SAP Data Services
Document Version: 4.2 Support Package 1 (14.2.1.0) - 2014-02-06

Technical Manuals

Table of Contents

1 Administrator Guide

1.1 Getting Started

1.1.1 Overview of SAP Data Services

About this section

This section introduces SAP Data Services and explains its place in the SAP solution portfolio.

1.1.1.1 SAP Data Services and the SAP solution portfolio

The SAP solution portfolio delivers extreme insight through specialized end-user tools on a single, trusted business intelligence platform. This entire platform is supported by SAP Data Services. On top of SAP Data Services, the SAP solution portfolio layers the most reliable, scalable, flexible, and manageable business intelligence (BI) platform which supports the industry's best integrated end-user interfaces: reporting, query and analysis, and performance management dashboards, scorecards, and applications.

True data integration blends batch extraction, transformation, and loading (ETL) technology with real-time bi-directional data flow across multiple applications for the extended enterprise.

By building a relational datastore and intelligently blending direct real-time and batch data-access methods to access data from enterprise resource planning (ERP) systems and other sources, SAP has created a powerful, high-performance data integration product that allows you to fully leverage your ERP and enterprise application infrastructure for multiple uses.

SAP provides a batch and real-time data integration system to drive today's new generation of analytic and supply-chain management applications. Using the highly scalable data integration solution provided by SAP, your enterprise can maintain a real-time, on-line dialogue with customers, suppliers, employees, and partners, providing them with the critical information they need for transactions and business analysis.

1.1.1.2 Software benefits

Use SAP Data Services to develop enterprise data integration for batch and real-time uses. With the software:

- You can create a single infrastructure for batch and real-time data movement to enable faster and lower cost implementation.
- Your enterprise can manage data as a corporate asset independent of any single system. Integrate data across many systems and reuse that data for many purposes.
- You have the option of using pre-packaged data solutions for fast deployment and quick ROI. These solutions extract historical and daily data from operational systems and cache this data in open relational databases.

The software customizes and manages data access and uniquely combines industry-leading, patent-pending technologies for delivering data to analytic, supply-chain management, customer relationship management, and Web applications.

1.1.1.2.1 Unification with the platform

SAP Data Services provides several points of platform unification:

- Get end-to-end data lineage and impact analysis.
- Create the semantic layer (universe) and manage change within the ETL design environment.

Data Services deeply integrates the entire ETL process with the business intelligence platform so you benefit from:

- Easy metadata management
- Simplified and unified administration
- Life cycle management
- Trusted information

1.1.1.2.2 Ease of use and high productivity

SAP Data Services combines both batch and real-time data movement and management to provide a single data integration platform for information management from any information source, for any information use.

Using the software, you can:

- Stage data in an operational datastore, data warehouse, or data mart.
- Update staged data in batch or real-time modes.
- Create a single graphical development environment for developing, testing, and deploying the entire data integration platform.
- Manage a single metadata repository to capture the relationships between different extraction and access methods and provide integrated lineage and impact analysis.

1.1.1.2.3 High availability and performance

The high-performance engine and proven data movement and management capabilities of SAP Data Services include:

- Scalable, multi-instance data-movement for fast execution
- Load balancing
- Changed-data capture
- Parallel processing

1.1.1.3 Associated software

Choose from other SAP solution portfolio software options to further support and enhance the power of your SAP Data Services software.

1.1.1.3.1 SAP Metadata Management

SAP BusinessObjects Metadata Management provides an integrated view of metadata and its multiple relationships for a complete Business Intelligence project spanning some or all of the SAP solution portfolio. Use the software to:

- View metadata about reports, documents, and data sources from a single repository.
- Analyze lineage to determine data sources of documents and reports.
- Analyze the impact of changing a source table, column, element, or field on existing documents and reports.
- Track different versions (changes) to each object over time.
- View operational metadata (such as the number of rows processed and CPU utilization) as historical data with a datetime.
- View metadata in different languages.

For more information on SAP Metadata Management, contact your SAP sales representative.

1.1.1.4 Interfaces

SAP Data Services provides many types of interface components. Your version of the software may provide some or all of them.

You can use the Adapter SDK to develop adapters that read from and/or write to other applications.

In addition to the interfaces listed above, the Nested Relational Data Model (NRDM) allows you to apply the full power of SQL transforms to manipulate, process, and enrich hierarchical business documents.

For a detailed list of supported environments and hardware requirements, see the *Product Availability Matrix* available at *https://service.sap.com/PAM*. This document includes specific version and patch-level requirements for databases, applications, web application servers, web browsers, and operating systems.

Related Information

Designer Guide: Nested Data [page 342]

1.1.2 Naming Conventions

In this documentation, the following naming conventions apply:

Terminology

- “Data Services system” refers to “SAP Data Services”.
- “BI platform” refers to “SAP BusinessObjects BI platform”.

  > Note
  >
  > The BI platform components required by Data Services may also be provided by SAP BusinessObjects Information platform services (IPS).

- “CMC” refers to the Central Management Console provided by the BI or IPS platform.
- “CMS” refers to the Central Management Server provided by BI or IPS platform.

Variables

Variables	Description
`<INSTALL_DIR>`	The installation directory for the SAP software. Default location: • `C:\Program Files (x86)\SAP BusinessObjects` • `$HOME/sap businessobjects`
`<BIP_INSTALL_DIR>`	The root directory of the BI or IPS platform. Default location: • `<INSTALL_DIR>\SAP BusinessObjects Enterprise XI 4.0` • `<INSTALL_DIR>/enterprise_xi40` Note These paths are the same for both the SAP BusinessObjects BI platform and SAP BusinessObjects Information platform services.
`<LINK_DIR>`	The root directory of the Data Services system. Default location: • All platforms `<INSTALL_DIR>/Data Services` This system environment variable is created automatically during installation.

Variables	Description
<DS_COMMON_DIR>	The common configuration directory for the Data Services system. Default location: - Windows (Vista and newer) `ALLUSERSPROFILE\SAP BusinessObjects\Data Services` - Windows (Older versions) `ALLUSERSPROFILE\Application Data\SAP BusinessObjects\Data Services` - UNIX systems (for compatibility) <LINK_DIR> This system environment variable is created automatically during installation.
<DS_USER_DIR>	The user-specific configuration directory for the Data Services system. Default location: - Windows (Vista and newer) `USERPROFILE\AppData\Local\SAP BusinessObjects\Data Services` - Windows (Older versions) `USERPROFILE\Local Settings\Application Data\SAP BusinessObjects\Data Services` This user environment variable is created automatically during installation. i Note This variable is used only for Data Services client applications on Windows, such as the Designer. <DS_USER_DIR> is not used on UNIX platforms.

1.2 Data Services Architecture

1.2.1 Architecture overview

This section outlines the overall platform architecture, system, and service components that make up the SAP Data Services platform. The information helps administrators understand the system essentials and help to form a plan for the system deployment, management, and maintenance.

Data Services is designed for high performance across a broad spectrum of user and deployment scenarios. For example:

- Developers can integrate Data Services into your organization's other technology systems by using web services, Java, or .NET application programming interfaces (APIs).
- End users can access, create, edit, and interact with Data Services projects and reports using specialized tools and applications that include:
  - Designer

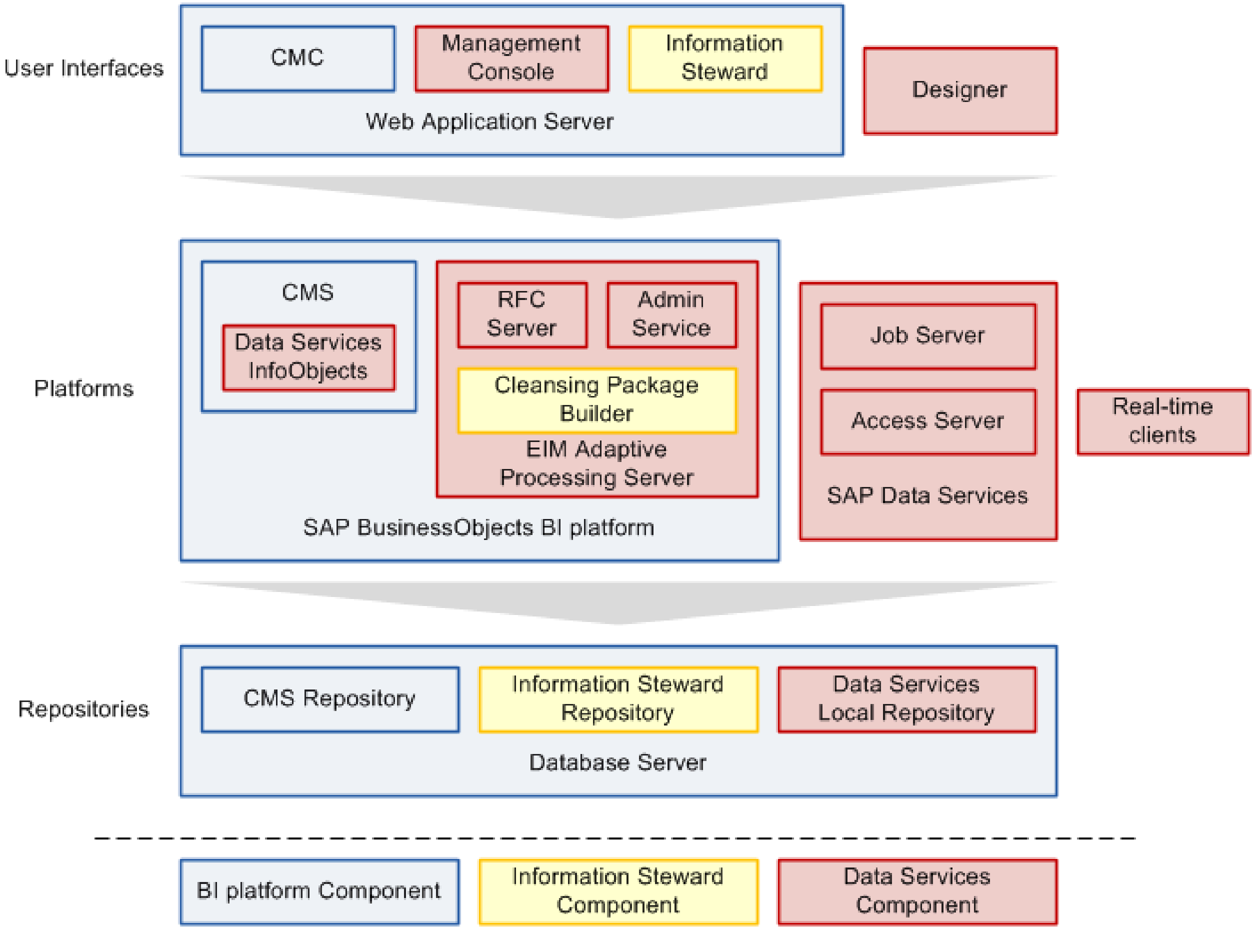
  - Management Console
    - Administrator
    - Impact and Lineage Analysis
    - Operational Dashboard
    - Auto Documentation
    - Data Validation
    - Data Quality
- IT departments can use data and system management tools that include:
  - Central Management Console (CMC)
  - Management Console
  - Server Manager
  - Repository Manager

To provide flexibility, reliability, and scalability, Data Services components can be installed on one or across many machines.

Server processes can be “vertically scaled” (where one computer runs several, or all, server-side processes) to reduce cost, or “horizontally scaled” (where server processes are distributed between two or more networked machines) to improve performance. It is also possible to run multiple, redundant versions of the same server process on more than one machine, so that processing can continue if the primary process encounters a problem.

1.2.2 Standard components

The following diagram illustrates how SAP Data Services components fit in with other software in the SAP portfolio.



Note

If you do not have a full SAP BusinessObjects BI platform installation, the basic components required by Data Services can also be provided by SAP BusinessObjects Information platform services.

For a detailed list of supported environments and hardware requirements, see the *Product Availablility Matrix* available at: *http://service.sap.com/PAM*. This information includes specific version and patch-level requirements for databases, applications, web application servers, web browsers, and operating systems.

1.2.2.1 Designer

The Designer is a development tool with an easy-to-use graphical user interface. It enables developers to define data management applications that consist of data mappings, transformations, and control logic.

Use the Designer to create applications containing work flows (job execution definitions) and data flows (data transformation definitions).

To use the Designer, create objects, then drag, drop, and configure them by selecting icons in flow diagrams, table layouts, and nested workspace pages. The objects in the Designer represent metadata. The Designer interface allows you to manage metadata stored in a repository. From the Designer, you can also trigger the Job Server to run your jobs for initial application testing.

Related Information

Repository [page 24]
Job Server [page 24]

1.2.2.2 Repository

The SAP Data Services repository is a set of tables that hold user-created and predefined system objects, source and target metadata, and transformation rules. Set up repositories on an open client/server platform to facilitate sharing metadata with other enterprise tools. Each repository must be stored on an existing RDBMS and registered in the Central Management Console (CMC).

Each repository is associated with one or more Job Servers which run the jobs you create. There are two types of repositories:

- Local repository
  A local repository is used by an application designer to store definitions of objects (like projects, jobs, work flows, and data flows) and source/target metadata.
- Central repository
  A central repository is an optional component that can be used to support multi-user development. The central repository provides a shared object library allowing developers to check objects in and out of their local repositories.
  While each user works on applications in a unique local repository, the team uses a central repository to store the master copy of the entire project. The central repository preserves all versions of an application's objects, so you can revert to a previous version if needed.
  Multi-user development includes other advanced features such as labeling and filtering to provide you with more flexibility and control in managing application objects.
  For more details, see the *Management Console Guide* and the *Designer Guide*.

1.2.2.3 Job Server

The SAP Data Services Job Server starts the data movement engine that integrates data from multiple heterogeneous sources, performs complex data transformations, and manages extractions and transactions from ERP systems and other sources. The Job Server can move data in either batch or real-time mode and uses distributed query optimization, multi-threading, in-memory caching, in-memory data transformations, and parallel processing to deliver high data throughput and scalability.

While designing a job, you can run it from the Designer which tells the Job Server to run the job. The Job Server retrieves the job from its associated repository, then starts an engine to process the job. In your production environment, the Job Server runs jobs triggered by a scheduler or by a real-time service managed by the Access Server. In production environments, you can balance job loads by creating a Job Server group (multiple Job Servers) which executes jobs according to overall system load.

Engine

When Data Services jobs are executed, the Job Server starts engine processes to perform data extraction, transformation, and movement. The engine processes use parallel processing and in-memory data transformations to deliver high data throughput and scalability.

Service

The Data Services service is installed when Job and Access Servers are installed. The service starts Job Servers and Access Servers when you restart your system. The Windows service name is `SAP Data Services`. The UNIX equivalent is a daemon named `AL_JobService`.

Related Information

Access Server [page 25]

1.2.2.4 Access Server

The SAP Data Services Access Server is a real-time, request-reply message broker that collects message requests, routes them to a real-time service, and delivers a message reply within a user-specified time frame. The Access Server queues messages and sends them to the next available real-time service across any number of computing resources. This approach provides automatic scalability because the Access Server can initiate additional real-time services on additional computing resources if traffic for a given real-time service is high. You can configure multiple Access Servers.

Service

The Data Services service is installed when Job and Access Servers are installed. The service starts Job Servers and Access Servers when you restart your system. The Windows service name is `SAP Data Services`. The UNIX equivalent is a daemon named `AL_JobService`.

1.2.2.5 Management Console

Administrator

The Administrator provides browser-based administration of SAP Data Services resources including:

- Scheduling, monitoring, and executing batch jobs.
- Configuring, starting, and stopping real-time services.
- Configuring Job Server, Access Server, and repository usage.
- Configuring and managing adapters.
- Managing users.
- Publishing batch jobs and real-time services via Web services.

Metadata Reports applications

The Metadata Reports applications provide browser-based analysis and reporting capabilities on metadata that is:

- Associated with your SAP Data Services jobs
- Associated with other SAP solution portfolio applications associated with Data Services

Metadata Reports provide several applications for exploring your metadata:

- Impact and lineage analysis
- Operational dashboards
- Auto documentation
- Data validation
- Data quality

1.2.2.5.1 Impact and Lineage Analysis reports

Impact and Lineage Analysis reports include:

- Datastore Analysis
  For each datastore connection, view overview, table, function, and hierarchy reports. SAP Data Services users can determine:
  - What data sources populate their tables
  - What target tables their tables populate
  - Whether one or more of the following SAP BusinessObjects solution portfolio reports uses data from their tables:
    - Business Views
    - Crystal Reports
    - SAP BusinessObjects Universe Builder
    - SAP BusinessObjects Web Intelligence documents
    - SAP BusinessObjects Desktop Intelligence documents
- Universe analysis
  View Universe, class, and object lineage. Universe users can determine what data sources populate their Universes and what reports use their Universes.
- Business View analysis
  View the data sources for Business Views in the Central Management Server (CMS). You can view business element and business field lineage reports for each Business View. Crystal Business View users can determine what data sources populate their Business Views and what reports use their views.

- Report analysis
  View data sources for reports in the Central Management Server (CMS). You can view table and column lineage reports for each Crystal Report and Web Intelligence Document managed by CMS. Report writers can determine what data sources populate their reports.
- Dependency analysis
  Search for specific objects in your repository and understand how those objects impact or are impacted by other SAP Data Services or SAP BusinessObjects Universe Builder objects and reports. Metadata search results provide links back into associated reports.

1.2.2.5.2 Operational Dashboard reports

Operational dashboard reports provide graphical depictions of SAP Data Services job execution statistics. This feedback allows you to view at a glance the status and performance of your job executions for one or more repositories over a given time period. You can then use this information to streamline and monitor your job scheduling and management for maximizing overall efficiency and performance.

1.2.2.5.3 Auto Documentation reports

Auto documentation reports provide a convenient and comprehensive way to create printed documentation for all of the objects you create in SAP Data Services. Auto documentation reports capture critical information for understanding your jobs so you can see at a glance the entire ETL process.

After creating a project, you can use Auto documentation reports to quickly create a PDF or Microsoft Word file that captures a selection of job, work flow, and/or data flow information including graphical representations and key mapping details.

1.2.2.5.4 Data Validation dashboard

Data Validation dashboard reports provide graphical depictions that let you evaluate the reliability of your target data based on the validation rules you created in your SAP Data Services batch jobs. This feedback allows business users to quickly review, assess, and identify potential inconsistencies or errors in source data.

1.2.2.5.5 Data Quality reports

Data Quality reports allow you to view and export Crystal Reports for batch and real-time jobs that include statistics-generating transforms. Report types include job summaries, transform-specific reports, and transform group reports.

1.2.2.6 Adapter SDK

The SAP Data Services Adapter SDK provides a Java platform for rapid development of adapters to other applications and middleware products such as EAI systems. Adapters use industry-standard XML and Java technology to ease the learning curve. Adapters provide all necessary styles of interaction including:

- Reading, writing, and request-reply from SAP Data Services to other systems.
- Request-reply from other systems to SAP Data Services.

1.2.3 Management tools

SAP Data Services has several management tools to help you manage your components.

1.2.3.1 License Manager

The License Manager displays the SAP Data Services components for which you currently have a license.

1.2.3.2 Repository Manager

The Repository Manager allows you to create, upgrade, and check the versions of local and central repositories.

1.2.3.3 Server Manager

The Server Manager allows you to add, delete, or edit the properties of Job Servers and Access Servers. It is automatically installed on each computer on which you install a Job Server or Access Server.

Use the Server Manager to define links between Job Servers and repositories. You can link multiple Job Servers on different machines to a single repository (for load balancing) or each Job Server to multiple repositories (with one default) to support individual repositories (separating test from production, for example).

The Server Manager is also where you specify SMTP server settings for the smtp_to email function.

Related Information

Reference Guide: To define and enable the smtp_to function [page 1670]

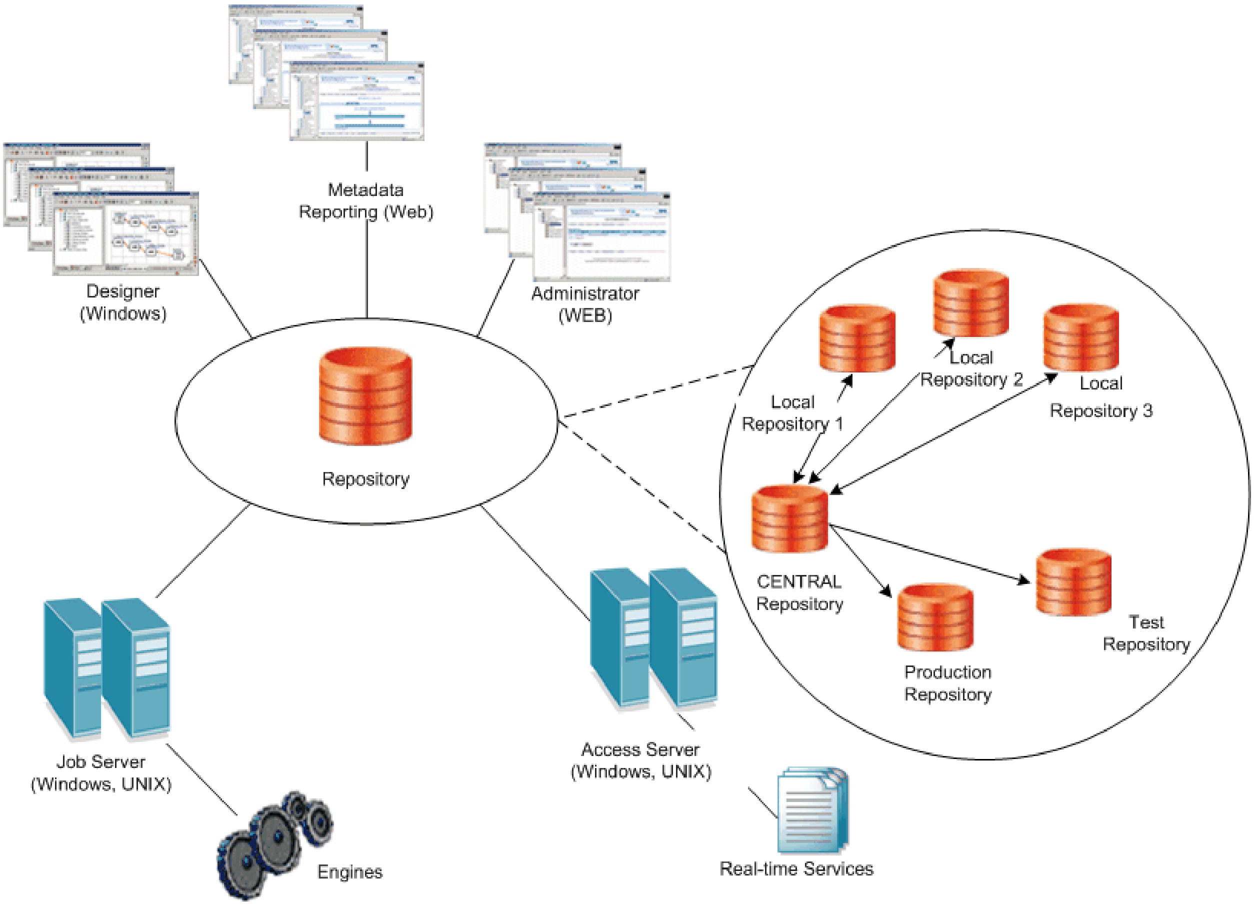
1.2.4 Operating system platforms

For a complete list of supported operating systems and hardware requirements, consult the *Product Availability Matrix* available at *http://service.sap.com/PAM*.

This document includes specific version and patch-level requirements for databases, applications, web application servers, web browsers, and operating systems.

1.2.5 Distributed architecture

SAP Data Services has a distributed architecture. An Access Server can serve multiple Job Servers and repositories. The multi-user licensed extension allows multiple Designers to work from a central repository. The following diagram illustrates both of these features.



You can distribute software components across multiple computers, subject to the following rules:

- Engine processes run on the same computer as the Job Server that spawns them.
- Adapters require a local Job Server.

Distribute components across a number of computers to best support the traffic and connectivity requirements of your network. You can create a minimally distributed system designed for developing and testing or a highly distributed system that can scale with the demands of a production environment.

1.2.5.1 Host names and port numbers

Communication between a Web application, the Access Server, the Job Server, and real-time services occurs through TCP/IP connections specified by IP addresses (or host names) and port numbers.

If your network does not use static addresses, use the name of the computer as the host name. If connecting to a computer that uses a static IP address, use that number as the host name for Access Server and Job Server configurations.

To allow for a highly scalable system, each component maintains its own list of connections. You define these connections through the Server Manager, the Administrator, the Repository Manager, and the Message Client library calls (from Web client).

For more information about the default port numbers used by Data Services, see the “Port assignments” section of the *Installation Guide*.

1.2.5.2 DSN-less and TNS-less connections

Data Services provides server name connections (also known as DSN-less and TNS-less connections) to databases that you use as a Data Services repository, source or target. Server name connections eliminate the need to configure the same DSN or TNS entries on every machine in a distributed environment.

For the Data Services repository, the following database types are supported:

- For Oracle databases, you specify the server name, database name, and port instead of the TNS name.
- For DB2, MySQL, and SAP HANA databases, you specify the server name, database name, and port instead of the DSN name.

> Note
>
> When you install Data Services, the repository defaults to a DSN-less or TNS-less connection. If you choose not to use a server name connection:
>
> - Defer repository creation to after installation.
> - Invoke the Repository Manager to subsequently create the repository.
>   - On Windows, select the option *Use TNS name* or *Use data source name (DSN)*.
>   - On UNIX, specify the `s` option to not use a server name connection.
> - Log in to the Central Management Console (CMC) to register the repository and select the repository connection type on the *Data Services Repository Properties* screen:
>   - For an Oracle database, select *Yes* in the drop-down list for *Use TNS name*.
>   - For a DB2, MySQL, or SAP HANA database, select *Yes* in the drop-down list for *Use data source name (DSN)*.

ℹ Note

This Data Services repository connection type setting on the CMC determines the connection type for logging into the Designer, running jobs, scheduling jobs, and so on.

For Data Services sources and targets, the following database types are supported for DSN-less and TNS-less connections:

- DB2 UDB
- Informix
- MySQL
- Netezza
- Oracle
- SAP HANA
- SAP Sybase IQ
- Teradata

ℹ Note

For the most current list of supported databases for server name connections, see the *Release Notes*.

Related Information

Administrator Guide: Using the Repository Manager [page 79]
Administrator Guide: To register a repository in the CMC [page 52]
Administrator Guide: To configure Job Servers [page 84]
Administrator Guide: To configure Job Servers on UNIX [page 91]
Administrator Guide: Using the Windows ODBC Driver Selector for DSN-less connections [page 79]
Administrator Guide: Configuring ODBC data sources on UNIX using DSN-less connections [page 69]

1.2.6 SAP integration

SAP Data Services integrates with your existing SAP infrastructure with the following SAP tools:

- SAP System Landscape Directory (SLD)
  The system landscape directory of SAP NetWeaver is the central source of system landscape information relevant for the management of your software life-cycle. By providing a directory comprising information about all installable software available from SAP and automatically updated data about systems already installed in a landscape, you get the foundation for tool support to plan software life-cycle tasks in your system landscape.
  The SAP Data Services installation program registers the vendor and product names and versions with the SLD, as well as server and front-end component names, versions, and location.
- SAP Solution Manager

The SAP Solution Manager is a platform that provides the integrated content, tools, and methodologies to implement, support, operate and monitor an organization's SAP and non-SAP solutions.
Non-SAP software with an SAP-certified integration is entered into a central repository and transferred automatically to your SAP System Landscape Directories (SLD). SAP customers can then easily identify which version of third-party product integration has been certified by SAP within their SAP system environment. This service provides additional awareness for third-party products besides our online catalogs for third-party products.
SAP Solution Manager is available to SAP customers at no extra charge, and includes direct access to SAP support and SAP product upgrade path information.

- CTS Transport (CTS+)
  The Change and Transport System (CTS) helps you to organize development projects in ABAP Workbench and in Customizing, and then transport the changes between the SAP systems in your system landscape. As well as ABAP objects, you can also transport Java objects (J2EE, JEE) and SAP-specific non-ABAP technologies (such as Web Dynpro Java or SAP NetWeaver Portal) in your landscape.
- Monitoring with CA Wily Introscope
  CA Wily Introscope is a web application management product that delivers the ability to monitor and diagnose performance problems that may occur within Java-based SAP modules in production, including visibility into custom Java applications and connections to back-end systems. It allows you to isolate performance bottlenecks in NetWeaver modules including individual Servlets, JSPs, EJBs, JCO's, Classes, Methods and more. It offers real-time, low-overhead monitoring, end-to-end transaction visibility, historical data for analysis or capacity planning, customizable dashboards, automated threshold alarms, and an open architecture to extend monitoring beyond NetWeaver environments.

1.3 Security

This section details the ways in which SAP Data Services addresses enterprise security concerns, thereby providing administrators and system architects with answers to typical questions regarding security.

Data Services relies on the Central Management Server (CMS) for authentication and security features. This section highlights differences and additional information specific to your Data Services system.

For complete information about the security features provided by the CMS, see the *SAP BusinessObjects BI Platform Administrator Guide* or the *SAP BusinessObjects Information Platform Services Administrator Guide*.

1.3.1 Securing administrative functions

To ensure security for your Data Services environment, use a firewall to prevent unintended remote access to administrative functions.

In a distributed installation, you need to configure your firewall so that the Data Services components are able to communicate with each other as needed.

For information about configuring ports on your firewall, see your firewall documentation. Also see the “Port assignments” topic in the Installation Guide

Related Information

Host names and port numbers [page 30]

1.3.2 Message client library

The Message Client libraries (Java and C++) used in real-time services, does not require authorization to connect. Therefore, it is important to use caution when using these libraries.

For more information about using the Message Client library, see the *SAP Data Services Integrator Guide*.

1.3.3 Temporary cache files

In Data Services, temporary cache files are generated for a variety of functions and operations. Profiling, joins, table comparison, sorting, `lookup`, and `group_by` are some examples. Because these files are not encrypted, by default, care should be taken when working with confidential or other sensitive data. Both pageable and persistent caches create data files that are not encrypted, by default.

Temporary file location

The temporary files that Data Services creates are stored in `%COMMON_DIR%/log/pCache/<repository_string>/`. These files can be secured using the appropriate permissions at the OS level.

Pageable cache and key data

The pageable cache option in a data flow stores data in temporary files that are removed automatically after a data flow finishes executing.

Persistent cache

Data Services provides a datastore called Persistent cache. The data in persistent cache is not encrypted, and it is your responsibility to secure it using OS file/directory permissions.

`long` data

When `long` data (BLOB or CLOB) data is large, the data is stored in temporary cache files.

If `long` data is cached (for a join, sort, or table comparison, for example), the cache file is deleted when the data flow finishes executing.

A `long` data cache file is also deleted when the data is out of scope. For example:

- The data is loaded into a target.
- The data is filtered out by a Query transform.
- A `long` datatype is converted to a `varchar`.

1.3.3.1 To encrypt certain temporary cache files

There are types of temporary cache files that can be encrypted, if necessary. These include:

- Persistent cache datastore files
- Pageable cache data flow files
- Functions such as lookup, search_replace, distinct, group_by, and so on.
- Transforms such as Data Quality transforms and Table Comparison

To enrypt these files:

1. Open the `DSConfig.txt` file, located in `%DS_COMMON_DIR%\conf`.
2. Set the `pageable_cache_encrypt_data` parameter, in the `String` section, to **`yes`**.
3. Save and close the file.

i Note

Encrypting these files can have a significant negative impact on performance. Remember that these files are deleted immediately after the data flow finishes executing.

1.3.4 Configuring SSL for Data Services components

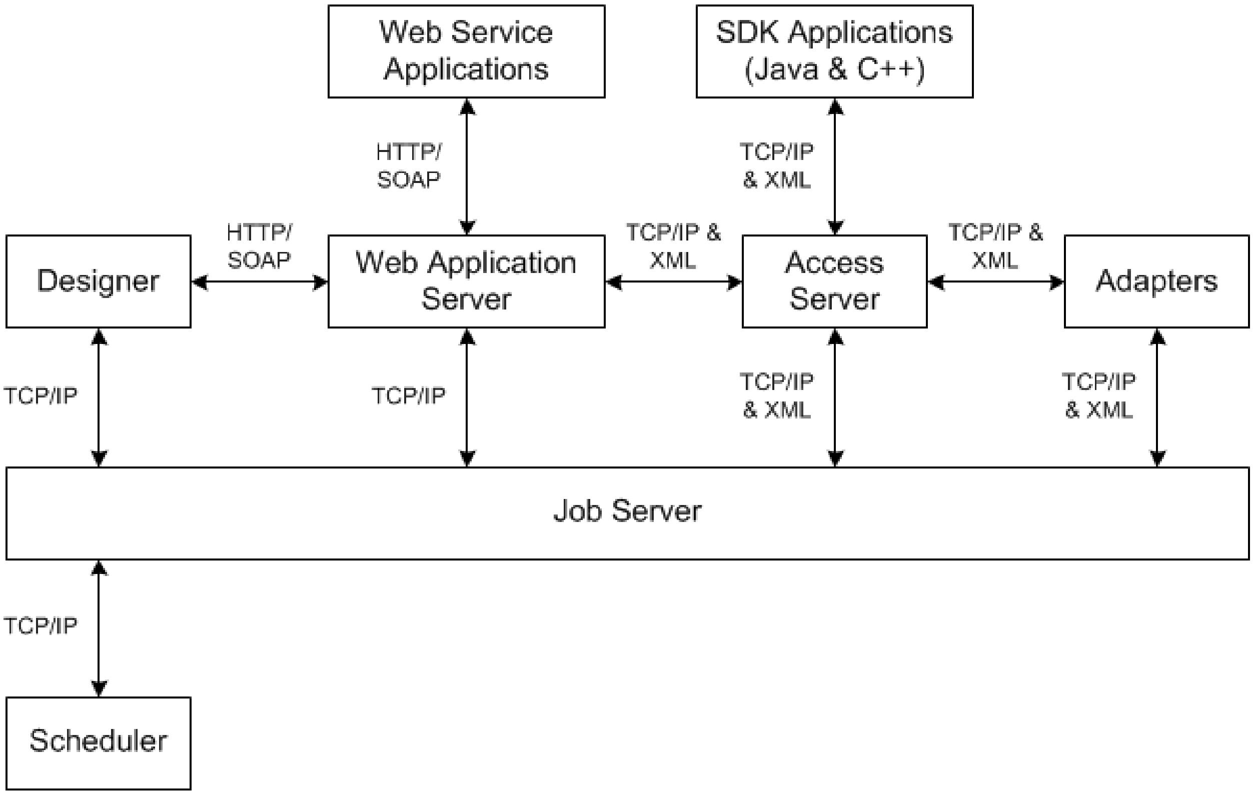
Secure Sockets Layer (SSL) is a cryptographic protocol that provides security and data integrity for communications over networks. Transport Layer Security (TLS) is the standard specification published by the IETF that is based on earlier SSL specifications.

The TLS protocol allows client/server applications to communicate across a network in a way designed to prevent eavesdropping, tampering, and message forgery. TLS provides endpoint authentication and communications confidentially over the network using cryptography.

Protected communication paths

Within the SAP Data Services platform, SSL is supported for all communication paths between components that communicate over a network.

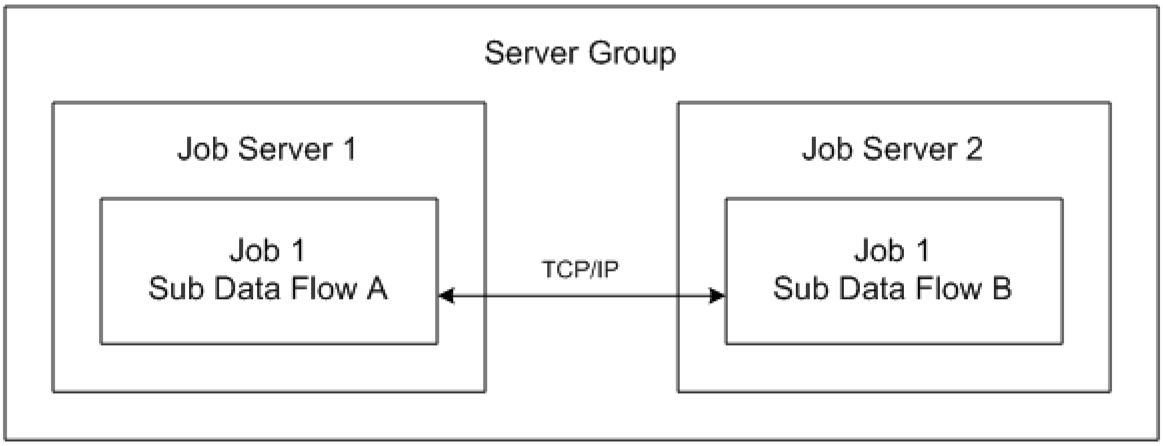
This diagram illustrates the communication channels within the Data Services architecture that support SSL.



Note

All TCP/IP communication paths support SSL/TLS. Depending on your web application server communication, clients using HTTP may switch to the HTTPS protocol.

Additionally, when you use a server group and set the distribution level to “Sub data flow”, the TCP/IP communication path between sub data flows on different job servers within the server group is also protected by SSL.



Default certificates

By default, a set of SSL certificates is created during installation for secure communication between Data Services components. You can choose to use your own certificates by configuring them after installation has finished. The default certificates use 1024-bit RSA keys and are valid for 30 years.

Related Information

To use custom certificates [page 39]
To copy certificates in a distributed installation [page 36]

1.3.4.1 To copy certificates in a distributed installation

When different Data Services components are installed on different machines and each installation has its own root and intermediate certificate authority (CA) configuration, you must manually copy the trusted certificates from one machine to all other machines.

Note

Trusted certificate files refers to root and intermediate CA certificate files. These files have a `.crt` extension, and can be located in the **`<LINK_DIR>`**`/ssl/trusted_certs` folder.

Remember

When you copy trusted certificates from one host machine to another, you must always copy the files to and from the **`<LINK_DIR>`**`/ssl/trusted_certs` folder on each respective machine.

1. If the Job Server and Access Server are installed on different machines, configure the hosts with the new certificates.
   a) Copy the trusted certificates from the Access Server to the Job Server host.
   b) On the Job Server host machine, run the following script to refresh the **<LINK_DIR>**`/ssl/trusted_certs/jssecacerts` keystore file:
      - On Windows: **<LINK_DIR>**`/bin/SetupJavaKeystore.bat`
      - On UNIX: **<LINK_DIR>**`/bin/SetupJavaKeystore.sh`

      This allows adapters that communicate with the Access Server to use the new certificates.
   c) Copy the trusted certificates from the Job Server to the Access Server host.
   d) Restart the job service on both the Job Server and Access Server host machines.
2. If the Access Server and Management Console are installed on different machines, configure the Management Console host with the new certificates.
   a) Copy the trusted certificates from the Access Server to the Management Console host.
   b) On the Management Console host machine, run the following script to refresh the **<LINK_DIR>**`/ssl/trusted_certs/jssecacerts` keystore file:
      - On Windows: **<LINK_DIR>**`/bin/SetupJavaKeystore.bat`
      - On UNIX: **<LINK_DIR>**`/bin/SetupJavaKeystore.sh`
   c) Restart the web application server that is hosting the Management Console.
3. If the Access Server and message client are installed on different machines, configure the message client host with the new certificates.
   a) Copy the trusted certificates from the Access Server to the message client host.
   b) If the message client uses Java, import the trusted certificates into the keystore used by the message client application.

      For information about creating keystores, see the JDK help for the `keytool` command.
4. If the Job Server and job launcher or external scheduler are installed on different machines, configure the job launcher or external scheduler host with the new certificates.

   Copy the trusted certificates from the Job Server to the job launcher or external scheduler host.

   > Note
   >
   > If the scheduled job connects to multiple Job Servers through a server group, copy the trusted certificates from all Job Servers within the group.

1.3.4.2 To enable or disable SSL on communication paths

Because Data Services uses multiple communication paths, there are different ways to enable or disable SSL for any given path. You may choose to enable or disable SSL for certain paths, depending on your security and performance requirements.

For adapter management

You can configure SSL for adapter management by enabling SSL support on your Job Servers. Enabling SSL for adapter management protects the communication path used between your Job Servers and adapters, and message broker clients.

To configure SSL on a Job Server, use the Server Manager.

For real-time messaging

You can configure SSL for real-time messaging by enabling SSL support on your Access Servers. Enabling SSL for real-time messaging protects the communication path used between your Access Servers and their real-time clients.

> ℹ Note
>
> By default, SSL is enabled for real-time messaging. If you disable it on an Access Server, be sure to disable it on any message clients or adapters that communicate with that Access Server.

> ℹ Note
>
> SSL can be enabled or disabled on a per-server basis. You are not required to configure it the same way for all Access Servers.

To configure SSL on an Access Server, use the Server Manager.

For peer-to-peer communication

You can configure SSL for peer-to-peer communication by configuring SSL for run-time resources. Enabling SSL for run-time resources protects the communication path used between different sub data flows running on different Job Servers.

> ℹ Note
>
> If you run multiple Job Servers within a server group, configure SSL the same way on each Job Server.

To configure SSL for run-time resources, use the Server Manager.

For other communication paths

SSL is mandatory for some communication paths within the Data Services architecture.

For example, SSL is always enabled on the communication paths between a Job Server and the following clients:

- The Administrator application in the Management Console
- Designers
- The job launcher
- Access Servers
- The job execution engine
- Other Job Servers within a server group
- The job service used for monitoring

You must ensure that each client has the correct certificates in these situations, but there is no additional configuration to perform.

> **Note**
>
> You need to copy the certificates from the Job Server to the Access Server, Management Console, and external job launcher hosts. In all other cases, the certificates are exchanged automatically.

Related Information

Using the Server Manager on Windows [page 84]
Using the Server Manager on UNIX systems [page 90]

1.3.4.3 To use custom certificates

While SAP Data Services includes a set of SSL certificates by default, you can also choose to use your own certificates. Depending on the nature of your Data Services deployment, not all steps below may be required.

1. Generate certificates as needed, and have them signed by a trusted certificate authority (CA).

   For more information, see the "To generate keys and sign certificates" section.
2. Copy all required certificates to the Data Services client machines.

   > **Note**
   >
   > Each Data Services client requires the certificates for all CAs in the certificate chain when validating the certificate of the Data Services server. The certificates within a certificate chain are called trusted certificates and must be present on the local machine. In most cases, the certificate chain is the same for all clients, and therefore the same certificates must be present on all client machines.
3. If you are using Java-based clients, use the JDK `keytool` utility to generate a keystore containing the trusted certificates.
4. Configure server certificate and keyfile paths with the Server Manager.
5. Configure certificates for the Designer.
   a) Choose *Tools* > *Options* within the Designer.
   b) Navigate to the *SSL* category under *Designer*.
   c) Specify the locations of the certificate file, the private key file, and the trusted certificates folder.

If you change any SSL options other than *Use SSL protocol for profiler*, you must restart both the Designer and any Data Services servers.

Related Information

To configure SSL paths [page 88]
To generate keys and sign certificates [page 40]

1.3.4.4 To generate keys and sign certificates

To use your own custom certificates for SSL security in Data Services, you must generate the certificates and have them signed by a trusted certificate authority (CA), such as VeriSign.

1. Generate the RSA key and certificate using the `openssl` tool.

```
openssl req -config <LINK_DIR>\ssl\conf\openssl.conf -new -newkey rsa:1024 -
nodes -keyout <mykey.pem> -out <myreq.pem>
```

   where **`<mykey.pem>`** is the name of the key file to generate, and **`<myreq.pem>`** is the name of the certificate file to generate.

   > i Note
   >
   > By default, `openssl` is installed to **`<LINK_DIR>`**`\bin`. For more information about available options and commands, see the openssl documentation.

2. Send the RSA private key and certificate files to your external CA.
3. After you receive the signed certificate from your CA, use the Server Manager to specify the path to the new certificate and private key file.

   > i Note
   >
   > Trusted certificates from an external CA must be in PEM format. The signed certificates should be copied to the **`<LINK_DIR>`**`\ssl\trusted_certs` directory.

Related Information

To configure SSL paths [page 88]
To configure SSL paths [page 96]

1.3.5 Configuring SSL for the CMS connection

You can use the Secure Sockets Layer (SSL) protocol for all network communications between SAP Data Services clients and the Central Management Server (CMS).

To set up SSL for all CMS communication, you need to perform the following steps:

- Deploy the SAP BusinessObjects BI platform or Information platform services with SSL enabled.
- Create key and certificate files for each machine in your deployment.
- Configure the location of these files in the Central Configuration Manager (CCM) and your web application server.

For Data Services, you also need to use the `sslconfig` utility configure all components that log into the CMS for SSL, including:

- Designer
- Job Servers
- External schedulers and the job launcher
- Management Console (if deployed to a different web application server than the SAP BusinessObjects BI platform or Information platform services web tier)

> Note
>
> For J2EE web application servers, configure SSL by modifying the startup script.

By default, the utility is installed in the following location:

- For Windows:
  `<INSTALL_DIR>``\SAP BusinessObjects Enterprise XI 4.0\win32_x86\sslconfig.exe`
- For UNIX:
  `<INSTALL_DIR>``/sap_bobj/enterprise_xi40/`**`<platform>`**`/boe_sslconfig`
  Where **`<platform>`** matches your UNIX platform.

For more information about using `sslconfig` and configuring the CMS and its clients for SSL, see “Configuring the SSL protocol” in the *SAP BusinessObjects BI Platform Administrator Guide* or the *SAP BusinessObjects Information Platform Services Administrator Guide*.

1.3.6 Configuring SSL for Metadata Browsing and View Data Services

You can use the Secure Sockets Layer (SSL) protocol for all network communications between the SAP Data Services backend engine and the following EIM Adaptive Processing Server services:

- Metadata Browsing Service
- View Data Service

Data Services provides these services, but they are used by other SAP software products, such as the Data Insight module of SAP Information Steward.

Data Services provides the following files by default:

- Keystore file
  The server side (Metadata Browsing Service or View Data Service) requires a Java Server keystore file which contains a single key and all the certificates that are part of the certificate chain involved in signing the key. Passwords for the keystore file and the key are also required.
  Data Services places the default keystore file and password files under the **<LINK_DIR>**`\ssl\mds` folder.
- Trusted Certificates
  These certificates are used for signing the key that is stored in the Java keystore used on the server side. The client side (the Data Services backend engine) uses these trusted certificates to communicate with the server.
  Data Services places the trusted certificates under **<LINK_DIR>**`\ssl\mds\trusted_certs` folder.

1.3.6.1 To configure SSL for Metadata Browsing and View Data Services

To enable and configure SSL communications for Metadata Browsing and View Data Services:

1. Log into the Central Management Console (CMC) as a user with administrative rights to the Data Services application.
2. Go to the “Applications” management area of the CMC.
   The “Applications” dialog box appears.
3. Right-click the Data Services application and select *Settings*.
   The “Settings” dialog box appears.
4. In the drop-down list for *Enable SSL communication for Metadata Browsing and View Data Services*, select “Yes”.
5. If you want to use the default keystore and certificates (that Data Services provides or that you generate using the Data Services tool), take the following steps:
   a) In the drop-down list for *Use Default SSL Settings*, select “Yes”.
   b) Click *Save*.
6. If you do not want to use the default keystore and certificates and generated your own outside of Data Services, take the following steps:
   a) Ensure that your keystore is a Java keystore file that contains a single key with all the certificates that are part of the certificate chain involved in signing the key. You must provide a password for the key and a password for the keystore file.
   b) Ensure that your keystore file exists in the **<LINK_DIR>**`\ssl\mds` folder and the corresponding certificate files are placed under **<LINK_DIR>**`\ssl\mds\trusted_certs` folder.
   c) If you have multiple Metadata Browsing Service or View Data Service instances associated with the same CMS server, you must copy the keystore and certificate files to all the machines where these instances are installed.
   d) In the drop-down list for *Use Default SSL Settings*, select “No”.
   e) In the *KeyStore File* box, enter the name of the KeyStore file that you want to use.
   f) Enter the KeyStore password.
   g) Enter the Key password.
   h) Click *Save*.
7. Restart the EIM.AdaptiveProcessingServer as follows:

a) Go to the “Servers” management area of the CMC
b) Expand the “Service Categories” node and select “Enterprise Information Management Services”.
c) Select “EIMAdaptiveProcessingServer” in the right pane.
d) Click *Action* > *Restart Server*.

1.3.6.2 To create a keystore file and certificates using the Data Services tool

While SAP Data Services provides a keystore file and set of SSL certificates for the Metadata Browsing Service and View Data Service, you can also create a new key and certificates using the Data Services tool.

To create a new keystore file and SSL certificates to be used as the default SSL settings for the Metadata Browsing Service and View Data Service:

1. Run the `MDSSetupJavaKeyStore` tool.
   a) In a command-line window, change directory to **`<LINK_DIR>`**`\bin`.

   ```
   cd <LINK_DIR>\bin
   ```

   b) Run "MDSSetupJavaKeyStore.bat "

   ```
   MDSSetupJavaKeyStore
   ```

   The `MDSSetupJavaKeyStore` tool creates the following files:
   - Keystore file `DSJavaKeyStore.keystore` in **`<LINK_DIR>`**`\ssl\mds` containing a single key and all the certificates that are part of the certificate chain involved in signing the key
   - File sslks.key in **`<LINK_DIR>`**`\ssl\mds` containing the key password
   - File sslstore.key in **`<LINK_DIR>`**`\ssl\mds` containing the keystore password
2. If you already configured and enabled SSL for Metadata Browsing Service and View Data Service, restart the EIM.AdaptiveProcessingServer.
   The restart picks up the new keystore and certificate files as the default ones if you selected “Yes” for the option *Use Default SSL Settings*.
3. If you have not yet configured SSL for these services, see *To configure SSL for Metadata Browsing and View Data Services* [page 42].

1.3.7 Password encryption

Within the SAP Data Services system, all passwords are encrypted using the AES algorithm with 128-bit keys.

1.3.7.1 Encryption key storage locations

Because passwords can be stored in multiple places within the Data Services system, an individual key is associated with each storage location.

Password location	Associated key location
Local repository	`REPOKEY` column in the `AL_VERSION` table
Central repository	`REPOKEY` column in the `AL_VERSION` table
Management Console	`admin.key` located in the same directory as `admin.xml`
Access Server	`AS.key` located in the same directory as `AS.xml`
Adapter SDK	**`<DS_COMMON_DIR>`**`/adapters/adapter.key`
`DSConfig.txt`	**`<DS_COMMON_DIR>`**`/conf/DSConfig.key`
Data Services-managed schedules	If the schedule uses a password file, the password is stored in the password file. If the schedule does not use a password file, the password is located in the job command line.
External scheduler command lines	If the schedule uses a password file, the password is stored in the password file. If the schedule does not use a password file, the password is located in the job command line.

⚠ Caution

For encryption keys that are stored in files, Data Services protects the security of the key file with strong OS permissions. For example, the software sets owner-only read & write access to the file (`chmod 600` on UNIX systems). You should also protect the key file by restricting user access to the server host machine when possible.

1.3.7.2 Encrypting passwords manually

In most instances, password encryption is handled automatically by the various Data Services applications and utilities. However, for some tasks, you may need to manually encrypt a password. For example, you may want to generate a data flow on the fly for use with the object creation XML toolkit. If your data flow contains a datastore that requires a password, it needs to be encrypted before you can import and run it successfully.

When you need to manually encrypt a password, you can use the `al_encrypt` command-line utility installed with the software.

Related Information

Password encryption [page 161]

1.3.8 Password protection for a Data Services repository

When you log in to the Data Services Designer or open a Data Quality report in the Management Console, by default, you are prompted to enter the user name and password for the Data Services repository you are accessing. You can turn off this default behavior by granting permissions in the BI Platform or Information platform services Central Management Console.

In the CMC, when you grant the *Allow user to retrieve repository password* right, the Data Services' repository password will be sent from the CMS to the client (Designer or Management Console: DQ reports). Although this password is encrypted, and the communication channel can be secured through SSL, sending passwords could pose a risk, and malicious users could obtain access to the password. You can selectively grant this right for repositories. For example, you may want to grant the right for development repositories but not for production repositories.

Related Information

Repository management [page 52]
Designer Guide: Logging into the Designer [page 171]
Management Console Guide: Data Quality reports [page 1994]

1.3.8.1 To set Data Services repository permissions in the CMC

Use the following steps to add permissions for users to automatically retrieve the Data Services repository password when logging on to the Designer and for accessing Data Quality reports.

1. On the Home page of the CMC, click *Data Services*.
2. On the left side of the CMC, select *Repositories*.
3. Choose ▶ *Manage* ▶ *Security* ▶ *User Security* ▶.
4. Select the *Data Services Designer Users* group (for Designer access) or the *Data Services Monitor Users* group (for Data Quality reports access), and then click the *Assign Security* button.
5. In the *Assign Security* window, click the *Advanced* tab.
6. Click *Add/Remove Rights*.
7. On the left of the *Add/Remove Rights* window, click *Application*, and select *Data Services Repository*.
8. Under *Specific Rights for Data Services Repository*, select *Granted* for either or both of the following options:
   - *Allow user to retrieve repository password*
   - *Allow user to retrieve repository password that user owns*
9. Click *OK*.

By following the preceding steps, you have given all users in the Data Services Designer Users group (or the Data Services Monitor Users group) permissions for all Data Services repositories.

Note

If you have a Data Services development or test repository, for example, to which you would like to restrict access, you can do this on a case-by-case basis. To do this, access the Add/Remove Rights window using the following steps:

1. On the Home page of the CMC, click *Data Services*.
2. On the left side of the CMC, select *Repositories*, and then select the repository that you want edit rights for.
3. Continue with step 3 above to complete your task.

1.4 User and rights management

SAP Data Services uses the Central Management Server (CMS) for user accounts and rights management.

This section covers information and procedures specific to administrating Data Services. For detailed information about user accounts and rights management, see the *SAP BusinessObjects BI Platform Administrator Guide* or the *SAP BusinessObjects Information Platform Services Administrator Guide.*

1.4.1 User management

In the *Users and Groups* management area of the Central Management Console (CMC), you can specify the details required for a user to access Data Services. In addition to creating a user account, you must also grant the user access to any repositories they need to work with.

By default, the Data Services installation program does not create any user accounts. You can use the CMC to create new user accounts, or assign existing user accounts to the Data Services group accounts.

For detailed information about creating user accounts, see “Managing Enterprise and general accounts” in the *SAP BusinessObjects BI Platform Administrator Guide* or the *SAP BusinessObjects Information Platform Services Administrator Guide*.

Related Information

Managing security settings for repositories in the CMC [page 53]

1.4.2 Group management

Groups are collections of users who share the same account privileges. Therefore, you may create groups that are based on department, role, or location. Groups enable you to change the rights for users in one place (a group) instead of modifying the rights for each user account individually. Also, you can assign object rights to a group or groups.

In the *Users and Groups* area of the Central Management Console (CMC), you can create groups that give a number of people access to the report or folder. This enables you to make changes in one place instead of modifying each user account individually.

In addition to the basic SAP BusinessObjects BI platform or Information platform services group accounts, Data Services includes several default group accounts:

Account name	Description
Data Services Administrator	Members of this group have access to all Data Services administrative functionality.
Data Services Multi-user Administrator	Members of this group are limited to managing secure central repositories. This role is a subset of the Data Services Administrator role. Multi-user administrators can: ● Add and remove secure central repositories. ● Manage users and groups. ● View secure central repository reports.
Data Services Monitor User	Members of this group have access limited to options available from the Status tabs. For example, a monitor user can abort batch jobs but cannot execute or schedule them. A monitor user can restart, abort, or shut down an Access Server, service, adapter instance, or client interface but cannot add or remove them.
Data Services Profiler Administrator	Members of this group are limited to managing profiler repositories. This role is a subset of the Administrator role. Profiler administrators can: ● Manage profiler tasks in any profiler repository. ● Manage the Profiler configuration.
Data Services Profiler User	Members of this group are limited to managing profiler tasks in the profiler repository that is configured for the user.
Data Services Operator	Members of this group have all Administrator privileges except they cannot modify repository, access, or CMS servers nor update datastore settings.
Data Services Designer	Members of this group have access to the Designer.

1.4.2.1 Detailed application rights

Application rights are assigned to each user group. The default application rights granted to each group are described in the following table.

> **i Note**
>
> The Data Services Administrator group account is granted access to all of the available Data Services application rights.

Right Name	Designer Users w/ View access	Designer Users w/ Full access	Monitor Users	Multi-user Administrator	Operator Users	Profiler Admin. Users	Profiler Users
Access to Administrator	X	X	X	X	X	X	X
Access to Auto Documentation	X	X	X	X	X	X	X
Access to Data Quality Reports	X	X	X	X	X	X	X
Access to Designer	X	X					
Access to Impact and Lineage	X	X	X	X	X	X	X
Access to Operational Dashboard	X	X	X	X	X	X	X
Access to Validation Dashboard	X	X	X	X	X	X	X
Administrator overview	X	X	X	X	X	X	X
Execute batch job		X			X		
Manage access server configurations							
Manage adapter configurations					X		
Manage batch job history		X			X		
Manage central repository groups				X			
Manage certification log configurations							

Right Name	Designer Users w/ View access	Designer Users w/ Full access	Monitor Users	Multi-user Administrator	Operator Users	Profiler Admin. Users	Profiler Users
Manage datastore and substitution param configurations							
Manage Object Promotion Configurations							
Manage Object Promotion Import							
Manage profiler configurations						X	
Manage real-time client interface status					X		
Manage real-time logs					X		
Manage real-time service status					X		
Manage real-time status					X		
Manage RFC client and server configurations					X		
Manage repository resource					X		
Manage server group configurations		X			X		
Manage status							

Right Name	Designer Users w/ View access	Designer Users w/ Full access	Monitor Users	Multi-user Administrator	Operator Users	Profiler Admin. Users	Profiler Users
interval configuration							
Manage webservices configurations					X		
View adapter status			X		X		
View batch job history	X	X	X		X		
View Data Quality sample data	X	X	X	X	X	X	X
View profiler status						X	X
View internal information in log		X		X		X	
View real-time client interface status			X		X		
View real-time logs			X		X		
View real-time service status			X		X		
View real-time status			X		X		
View RFC client status			X		X		
View server group information	X	X	X		X		
View Validation sample data	X	X	X	X	X	X	X
View webservices status			X		X		

Related Information

Viewing application rights assigned to a group [page 51]
Managing application rights for a group [page 51]

1.4.2.2 Viewing application rights assigned to a group

Your account must be a member of the Administrators user group or a member of the Data Services Administrator Users group to be able access the CMC function for managing user security.

To view a group's currently assigned application rights:

1. On the Home page of the CMC, select *Manage > Applications*.
2. In the *Application Name* list, double-click *Data Services Application*.
3. Select *User Security* to display the user groups.
4. Highlight the user group you want to view and click *View Security*.

The *Permissions Explorer* opens and displays all of the current application rights that are assigned to your selected user group.

1.4.2.3 Managing application rights for a group

Your account must be a member of the Administrators user group or a member of the Data Services Administrator Users group to be able access the CMC function for managing user security.

To manage a group's assigned application rights:

1. On the Home page of the CMC, select *Manage > Applications*.
2. In the *Application Name* list, double-click *Data Services Application*.
3. Select *User Security* to display the configured Data Services application user groups.
4. Highlight the user group or add a user to manage their application rights.
5. Click *Assign Security*.
6. In the *Assign Security* window, select the *Advanced* tab.
7. On the *Advanced* tab, select *Add/Remove rights*.
8. In the *Add/Remove Rights* window, select to add or remove each of the specific right you want to change for this group.
9. Click *Apply* to save your changes.

Members of the modified group will be granted or denied the application rights you modified for the group the next time they log in.

1.5 Repository management

Before you can access Data Services repositories in other components such as the Designer and Management Console, you must configure them appropriately.

In general, you follow this work flow to configure a Data Services repository.

1. Create the repository in a database using the Repository Manager.
2. Register the repository in the Central Management Console (CMC).
3. Manage security settings for the repository in the CMC.

1.5.1 To register a repository in the CMC

1. Log into the Central Management Console (CMC) as a user with administrative rights to the Data Services application.
2. Go to the Data Services application:
   - Click *Data Services* from the CMC home screen OR
   - Click the Data Services icon
3. Configure a repository:
   - Choose ▶ *Manage* ▶ *Configure Repository* ◀ OR
   - Right-click *Repositories* in the navigation tree and click *Configure Repository*

   The *Add Data Services Repository* screen is displayed.
4. Specify a name and optionally a description for the repository.

   The name and description will be displayed to users when they log into applications such as the Designer and Management Console.
5. Enter the connection information for the repository database.

   The details required depend on the type of database containing the repository and the connection type you choose.

   > ➜ Tip
   >
   > For Microsoft SQL Server and SAP Sybase databases, it is recommended that you do not use `localhost` as the server name when registering the Data Services repository in the CMC. If you use `localhost`, other machines will be unable to connect to the repository.

   a) For an Oracle database, the default connection type is TNS-less. If you want to use a TNS connection, select *Yes* in the drop-down menu for *Use TNS Name* and enter the *TNS Name* if no value appears.

   If the Data Services repository was created using a TNS connection, the software fills in *TNS Name*. Otherwise, you must enter the *TNS Name*.

   > i Note
   >
   > If you created a repository on Oracle RAC, prior to registering in the CMC, you need to configure TNS Name on the local CMS machine. Then the TNS name will be filled in automatically based on the connection string provided when you register in the CMC .

b) For a DB2, MySQL or SAP HANA database, the default connection type is DSN-less. If you want to use a DSN connection, select *Yes* in the drop-down menu for *Use Data Source Name (DSN)* and enter the *Data Source Name (DSN)* if no value appears.

   If the Data Services repository was created using a DSN connection, the software fills in *Data Source Name (DSN)*. Otherwise, you must enter the *Data Source Name (DSN)*.

> i Note
>
> If you subsequently edit the properties of the Data Services repository on the CMC to change the connection type to a TNS or DSN connection, you must fill in *TNS Name* or *Data Source Name (DSN)*.

> i Note
>
> If you are using DNS or TNS connections in a distributed installation, the database connection to the repository must be configured the same on each machine as in the CMC. For example, if an Oracle repository is configured with the TNS name `Ora_DS` in the CMC, Designer and Job Server machines must also have the `Ora_DS` TNS name configured.

6. If you are registering a profiler repository, choose *Yes* for *Is Profiler Repository*.
   a) Enter the host name of the web application server hosting the profiler.
   b) Enter the port number used by the web application server hosting the profiler.
7. Click *Test Connection*.
   The application attempts to verify the connection details without adding the repository to the CMC. If the connection is not successful, review the error message and correct the repository connection information.
8. Click *Save* to add the repository to the CMC.

The Data Services application screen is displayed, and the new repository appears in the list of registered repositories.

Related Information

DSN-less and TNS-less connections [page 30]
Using the Repository Manager [page 79]

1.5.2 Managing security settings for repositories in the CMC

You can manage security settings for repositories registered in the CMC with the security options on the Manage menu. These options let you assign users and groups to the access control list for the repository, view the rights that a user or group has, and modify the rights that the user or group has to the repository.

1. Log into the Central Management Console (CMC) as a user with administrative rights to the Data Services application.
2. Navigate to the Data Services application:
   ○ Click *Data Services* on the CMC home screen OR

- Click the Data Services icon

3. Click *Repositories* in the navigation tree to display the list of registered repositories.

1.5.2.1 To view rights for a user or group on a repository

In general, you follow this work flow to view rights for a user or group on a repository.

1. Select the repository for which you want to view security settings.
2. Click *Manage* > *Security* > *User Security*.
   The *User Security* dialog box appears and displays the access control list for the repository.
3. Select a user or group from the access control list, and click *View Security*.
   The Permissions Explorer launches and displays a list of effective rights for the user or group on the repository.

1.5.2.2 To assign users and groups to an access control list for a repository

An access control list specifies the users that are granted or denied rights to a repository. In general, you follow this work flow to assign a user or group to an access control list, and to specify the rights that the user or group has to the repository.

1. Select the repository to which you want to add a user or group.
2. Click *Manage* > *Security* > *User Security*.
   The *User Security* dialog box appears and displays the access control list for the repository.
3. Click *Add Principals*.
   The *Add Principals* dialog box appears.
4. Move the users and groups you want to add from the *Available users/groups* list to the *Selected users/groups* list.
5. Click *Add and Assign Security*.
6. Select the access levels you want to grant the user or group:
   - To grant read-only access to the repository, select *View*.
   - To grant full read and write access to the repository, select *Full Control*.
   - To deny all access to the repository, select *No Access*.

1.6 Server management

1.6.1 Setting UNIX environment variables

When you install SAP Data Services on UNIX platforms, the Job Server requires that certain environment variables be set up. To set up these variables, users who run or administer Job Servers must run a script (al_env.sh).

Run this script with the syntax required by your environment. For example:

```
$ cd $LINK_DIR/bin/
$ . ./al_env.sh
```

You can also add this command to your login script so that it is always configured. For example, add the following line to the .profile:

```
. $LINK_DIR/bin/al_env.sh
```

If the script fails to run, no error messages appear. To make sure that the variables' values are properly set, check one or more of the following:

Variable	Details
$LINK_DIR	Data Services installation directory (set by the installation program).
$DS_COMMON_DIR	References $LINK_DIR for compatibility (set by the installation program).
$SHLIB_PATH	If you want to use a 64-bit Oracle client, $LINK_DIR/bin must be listed before any 64-bit Oracle shared library path.
$LD_LIBRARY_PATH	For Solaris or Linux. Must include $LINK_DIR/bin and the location of the database libraries. If you want to use a 64-bit Oracle client, $LINK_DIR/bin must be listed before any 64-bit Oracle shared library path.
$LIBPATH	For AIX. Must include $LINK_DIR/bin and the location of the database libraries. If you want to use a 64-bit Oracle client, $LINK_DIR/bin must be listed before any 64-bit Oracle shared library path.
$ORACLE_SID	Required for an Oracle source, target, or repository.
$ORACLE_HOME	Required for an Oracle source, target, or repository. If you want to use a 64-bit Oracle client, this must point to the 64-bit Oracle installation.
$DB2INSTANCE	Required for a DB2 source, target, or repository.
$DB2DIR	Required for a DB2 source, target, or repository.
$SYBASE	Required for a SAP Sybase ASE source, target, or repository.
$SYBASE_OCS	Required for a SAP Sybase ASE source, target, or repository.

Variable	Details
`$ODBCINI`	Required for ODBC sources or targets, including MySQL and SAP HANA.
`$PATH`	Must include `$LINK_DIR/bin` and **`<databasehome>`**`/bin`.

➜ Tip

Use the `echo` command to verify environment variable settings.

If the variable settings are not properly configured and you start any Data Services utility, error messages indicate that database server files are missing.

If you see such an error, verify that `al_env.sh` contains commands to set the appropriate database home locations. Run `al_env.sh` for the account used by the Job Server, or start the Job Server using an account that has all necessary environment variables defined in its `.profile`.

➜ Tip

If you want to use the RFC Server Interface in the Management Console on a 64-bit UNIX platform, see the *Management Console Guide* for additional environment configuration information.

1.6.1.1 Configuring additional database connectivity

When you install SAP Data Services on UNIX platforms, the installation setup program configures the following by default:

- DSN or TNS connections for the repository database
  To use a DSN-less or TNS-less connection, defer repository creation to after installation and when you subsequently invoke the Repository Manager, specify the `s` option to use a server name connection.
- Database connectivity for the repository only
  To access other database systems as sources and targets in your jobs, you need to add the appropriate configuration information to the `al_env.sh` file. Use the Connection Manager that is installed with Data Services to set the environment variables required for the following database types:
  - Attunity
  - DB2 on iSeries or zSeries
  - Informix
  - MySQL
  - Netezza
  - Oracle
  - SAP HANA
  - SAP Sybase ASE
  - SAP Sybase IQ
  - SAP Sybase SQL Anywhere
  - SQL Server
  - Teradata

Note

For the most current list of databases types supported by the Connection Manager, see the Supported Platforms (Product Availability Matrix) *https://service.sap.com/PAM*.

Related Information

Using the Connection Manager for UNIX systems [page 60]
Configuring ODBC data sources on UNIX using DSN connections [page 61]
Configuring ODBC drivers on UNIX for data sources using DSN-less connections [page 69]

1.6.2 Starting services automatically

On Windows

The SAP Data Services service and packaged Tomcat service start automatically when the computer restarts. The Data Services service then starts Job Servers and Access Servers on the restarted computer.

You can change service startup to *Manual* in the Windows services window.

Note

To manually log in Web applications, refer to “Configuring tracing for web applications” section in the *Information platform services Administration Guide.*

On UNIX

To start Job and Access Servers automatically when the server restarts, you must install the `actaservices` script with root privileges.

Run the `$LINK_DIR/bin/autostrt.sh` script:

```
# cd $LINK_DIR/bin/
# autostrt.sh $LINK_DIR
```

1.6.3 Setting the log retention period

The log retention period provides an automatic way to delete log files. You can view currently stored logs with the Administrator application in the Data Services Management Console .

Follow these steps to set the job server log retention period:

1. Open the Central Management Console (CMC) in a web browser and log on as a user with administrative rights.
2. Choose *Applications* from the navigation drop-down menu under the *Central Management Console* banner.
3. Right-click *Data Services Application* from the *Application Name* column and select *Settings*.
4. In the *Job Server Log Retention Period* box, type the number of days that you want to retain the following:
   - Historical batch job error, trace, and monitor logs.
   - Current service provider trace and error logs.
   - Current and historical Access Server logs.

   The software deletes all log files beyond this period. For example:

Enter	Results
1	The software displays the logs for today only. After 12:00 AM these logs clear and the software starts saving logs for the next day.
0	The software does not retain any log files.
-1	The software does not delete any log files.
1095	The software deletes log files older than approximately three years.

5. Click *Save*.

Changes you make to the log retention period occur as a background clean-up process so they do not interrupt more important message processing. Therefore, logs might not be deleted immediately when you select *Save*. Changes can take up to an hour to take effect.

For more information about viewing log files in the Administrator, see the *Management Console Guide*.

Related Information

Setting the history retention period [page 58]
Designer Guide: DSF2 Augment Statistics log files [page 650]
Designer Guide: NCOALink logs files [page 664]
Management Console Guide: Data Quality Reports [page 1994]

1.6.4 Setting the history retention period

The log retention period provides an automatic way to delete log files. You can view currently stored logs with the Administrator application in the Data Services Management Console.

Follow these steps to set the *History Retention Period*:

1. Open the Central Management Console (CMC) in a web browser and log on as a user with administrative rights.

2. Choose *Applications* from the navigation drop-down menu under the *Central Management Console* banner.
3. Right-click *Data Services Application* from the *Application Name* column and select *Settings*.
4. In the *History Retention Period* box, type the number of days that you want to retain job execution history, which includes the following information:
   - Certification and non-certification log files.
   - Report information.

   The software deletes all log files beyond this period. For example:

Enter	Results
1	The software displays the jobs executed today only. After 12:00 AM these logs clear and the software starts saving logs for the next day.
0	The software does not retain any job history files.
-1	The software does not delete any job history files.
1095	The software deletes job history older than approximately three years.

5. Click *Save*.

Related Information

Setting the log retention period [page 57]
Designer Guide: DSF2 Augment Statistics log files [page 650]
Designer Guide: NCOALink logs files [page 664]
Management Console Guide: Data Quality Reports [page 1994]

1.6.4.1 USPS-required log files and reports

If you have postal certification requirements such as CASS certification, you are required to submit log files to the postal authorities on a periodic basis. For example, if you have included the USA Regulatory Address Cleanse transform in your data flow, and you use the DSF2 and/or NCOALink functionality and have CASS enabled, you must submit certification log files to the USPS each month. With that in mind, set the log retention period so that you will not lose data prior to the creation and submission of the logs (thus submitting incomplete log files to the USPS).

The default setting of 30 days does not provide enough time for you to export and send the log files to the USPS by the monthly due date. And 30 days does not account for months that include 31 days. Therefore we recommend setting the history retention to longer (50 days for example) to ensure that you submit complete monthly logs to the USPS.

Set the number of retention days in the history retention period setting in the CMC.

In addition to sending monthly data to the USPS, you are required to have report data available for the USPS to examine for several years after the job is processed. (Make sure you are aware of current USPS rules for data retention by viewing your USPS licensing agreement.) To ensure that you retain all required reports and logs before the data is deleted from the repository, we recommend that you export the required reports and logs from the repository to a local folder on a monthly basis. This also prevents the repository contents from becoming so large that the export process “times out” due to the volume of statistics retained.

Related Information

Set history retention period [page 58]
Designer Guide: DSF2 Augment Statistics log files [page 650]
Designer Guide: NCOALink logs files [page 664]

1.6.5 Using the Connection Manager for UNIX systems

Use the Connection Manager on UNIX to create, edit, or delete ODBC data sources and ODBC drivers after installation.

1. If you want to use the graphical user interface, ensure you have installed the GTK+2 library.
2. For DSN connections, set `$ODBCINI` to a file that will define the DSN. Ensure that the file is readable and writeable.
3. Invoke the Connection Manager by entering the following commands:

   ```
   $ cd $LINK_DIR/bin/
   $ ./DSConnectionManager
   ```

   The *SAP Data Services Connection Manager* window opens.

   > Note
   >
   > If the GTK+2 library is not installed, the command-line user interface starts.

4. For DSN connections, go to the *Data Sources* tab and configure data sources as needed. See the following sections for specific data sources.
5. For server name connections, go to the *Drivers* tab to configure ODBC drivers.
6. Click *Restart Services* to restart the EIM Adaptive Process Service and Data Services Job Service.

Related Information

Using the ODBC Driver Selector on Windows for server name connections [page 79]

1.6.5.1 Configuring ODBC data sources on UNIX using DSN connections

On UNIX and Linux platforms, SAP Data Services requires an ODBC driver manager library and ODBC driver library to configure ODBC data sources using data source name (DSN) connections. Some ODBC driver vendors include ODBC driver manager and ODBC driver capabilities in one single library, while others separate them into two individual libraries.

To accommodate all ODBC vendors, Data Services requires configurations in two different files for DSN connections:

1. The UNIX ODBC driver manager configuration file (**`<LINK_DIR>`**`/bin/ds_odbc.ini`).
   This file contains DSN instances that reference ODBC driver manager libraries.

   > ➡ Tip
   >
   > For natively supported ODBC databases, you do not need to manually configure `ds_odbc.ini`. Instead, use the Data Services Connection Manager to properly configure the ODBC driver manager library.

2. The ODBC vendor's configuration files (referenced by the `$ODBCINI` environment variable).
   This file contains DSN instances that reference the ODBC driver libraries, as well as the database server connection information.

   > i Note
   >
   > One ODBC configuration file can contain multiple different DSN instances. For example, a file referenced by `$ODBCINI` may contain DSNs for MySQL, Netezza, and Teradata.

1.6.5.1.1 Configuring native ODBC data sources

Data Services supports several ODBC data sources natively with DSN connections, including:

- Attunity
- DB2 on iSeries or zSeries
- Informix
- MySQL
- Netezza
- SAP HANA
- SAP Sybase ASE
- SAP Sybase IQ
- SQL Server
- Teradata

> i Note
>
> For the most current list of natively supported ODBC data sources, see the *Release Notes*.

You can use the Connection Manager to set the Data Services ODBC configuration and associated environment variables required to run a Data Services job that contains a source or target that is one of the above database types. Other generic ODBC data sources require additional manual configuration.

To run the Connection Manager to configure an ODBC source with a DSN connection:

1. Set `$ODBCINI` to a file in which the Connection Manager will define the DSN according to your input on the *Data Sources* tab. Ensure that the file is readable and writable.
   For example:

   ```
   export ODBCINI=<dir-path>/odbc.ini
   touch $ODBCINI
   ```

2. Invoke the Connection Manager by entering the following command:

   `$LINK_DIR/bin/DSConnectionManager.sh`
3. Click the *Data Sources* tab, and click *Add* to display the list of database types.
4. On the *Select Database Type* window, select the database type and click *OK*.
   The *Configuration for...* window appears with some of the connection information filled in with information that the Connection Manager detected:

   - The absolute location of the `odbc.ini` file in which the DSN will be defined
   - Driver (if relevant for database type)
   - Driver Version (if relevant for database type)
5. Provide values for additional connection properties (such as Server Name, Instance, or Port) for the specific database type.

   For a list of relevant properties for each database type, see *Properties for ODBC data sources using DSN connections* [page 63].
6. Provide the following properties (they will not be saved for further use).

   - User name
   - Password
7. If you want to test the connection, click *Test Connection*.
8. Click *Restart Services* to restart the following services:

   Restart Services

   - Both the EIM Adaptive Process Service and Data Services Job Service if Data Services is installed at the same location (machine and folder) as Information Platform Services (IPS) or BI platform. A prompt will appear for the CMS password.
   - Only the Data Services Job Service if Data Services is installed without IPS or BI platform.
9. If you will run another command such as the Repository Manager, source the `al_env.sh` script to set the environment variables.

   By default, the script is located at **`<LINK_DIR>`**`/bin/al_env.sh`.

Related Information

Configuring other ODBC data sources [page 72]
Properties for ODBC data sources using DSN connections [page 63]
The Connection Manager configures the `$ODBCINI` file based on the property values that you enter on the *Data Sources* tab. The following table lists the properties that are relevant for each database type.

1.6.5.1.2 Properties for ODBC data sources using DSN connections

The Connection Manager configures the `$ODBCINI` file based on the property values that you enter on the *Data Sources* tab. The following table lists the properties that are relevant for each database type.

Database Type	Properties on Data Sources tab
MySQL	• ODBC Ini File • DSN Name • Unix ODBC Lib Path • Driver • Driver Version • Server Name • Port • Database • User Name • Password
SQL Server	• ODBC Ini File • DSN Name • Server Name • Port • Database • User Name • Password
SAP HANA	• ODBC Ini File • DSN Name • Driver • Server Name • Instance • User Name • Password
DB2 on iSeries or zSeries	• ODBC Ini File • DSN Name • Server Name • Port • Location • Collection • Package Collection • User Name • Password
Teradata	• ODBC Ini File

Database Type	Properties on Data Sources tab
	• DSN Name • Teradata Install Path • Teradata Version • Server Name • User Name • Password
Netezza	• ODBC Ini File • DSN Name • Driver • Driver Version • Server Name • Port • Database • User Name • Password
Sybase IQ	• ODBC Ini File • DSN Name • Driver • Server Name This is also known as the host name. For example, a host name may be: vanpgc13b9 • Port • Engine Name This is also known as the server name. For example, a server name may be: vanpgc13b9_iqdemo • Database • User Name • Password
Sybase ASE	• Sybase Home Path • OCS • Server Name • Database • User Name • Password
Informix	• ODBC Ini File • DSN Name • Driver • Server Name • Database • User Name

Database Type	Properties on Data Sources tab
	● Password
Attunity	● Attunity Driver Path

1.6.5.1.3 To configure MySQL ODBC for DSN connections

Run the Connection Manager to set the Data Services ODBC configuration and associated environment variables required to run a Data Services job that contains a MySQL source or target.

1. Follow the same steps as in *Configuring native ODBC data sources* [page 61].
2. The MySQL ODBC connector driver (`libmyodbc<version>.so/sl`) has a dependency on the unixODBC driver manager(`libodbc.so`) provided by *www.unixodbc.org* .
   a) If you do not already have the unixODBC driver manager on your system, you must acquire and build the driver manager to resolve this dependency.
   b) Make sure you have the directory location of `libodbc.so` from the unixODBC installation as the first directory in the beginning of `LD_LIBRARY_PATH/LIBPATH/SHLIB_PATH`.

> ⚠ Caution
>
> If the first directory in `LD_LIBRARY_PATH/LIBPATH/SHLIB_PATH` has `libodbc.so` from a location other than the unixODBC installation, a job using MySQL as source/target/repository may not work as expected.

1.6.5.1.3.1 To install the unixODBC driver for Linux

To install the unixODBC driver, you must be using a version of Linux supported by SAP Data Services.

There are two ways to install the unixODBC libraries on Linux:

1. Install the bundled rpm unixODBC package on the Linux installation CD. For example, the the rpm package name on Redhat 5 64-bit is `unixODBC-2.2.11-7.1`.
2. Download and install the 64-bit unixODBC (x86_64) package from the following location: *http://sourceforge.net/projects/unixodbc/files/unixODBC/2.2.14/unixODBC-2.2.14-linux-x86-64.tar.gz/download*

For the latest supported versions, refer to the *Product Availability Matrix* available at *http://service.sap.com/PAM* .

1.6.5.1.3.2 To build and install the unixODBC driver for AIX

To install the unixODBC driver, you must be using a version of AIX supported by SAP Data Services, have the VisualAge C++ compiler (version 6.0 or greater), and download the unixODBC source.

1. Download and extract the unixODBC package.
   a) Download `unixODBC-2.2.12.tar.gz` from *http://www.unixodbc.org* to the `$TEMP` directory.

      > **Note**
      >
      > The downloaded file will be named `unixODBC-2.2.12.tar.tar`.

   b) Rename the unixODBC package.

      ```
      mv unixODBC-2.2.12.tar.tar unixODBC-2.2.12.tar.gz
      ```

   c) Extract the package with `gunzip` and `tar`.

      ```
      gunzip unixODBC-2.2.12.tar.gz
      tar -xvf unixODBC-2.2.12.tar
      ```

   d) Change to the newly created `unixODBC-2.2.12` directory.

      ```
      cd $TEMP/unixODBC-2.2.12
      ```

2. Make the libraries and programs.
   a) Ensure that xlC (the C++ compiler) is in the `PATH` environment variable, and add it if necessary.
      - Using ksh:

        ```
        export PATH=/usr/vacpp/bin:$PATH
        ```

      - Using csh:

        ```
        setenv PATH /usr/vacpp/bin:$PATH
        ```

   b) Configure the C compiler to be thread-enabled:

      ```
      export CC=xlc_r
      export CCC=xlC_r
      ```

   c) To compile a 64-bit version of the driver manager using the `xlc_r` compilers, set the `OBJECT_MODE` and `CFLAGS` environment variables:

      ```
      export OBJECT_MODE=64
      export CFLAGS=-q64
      ```

   d) Build the package.

      ```
      ./configure --enable-gui=no --enable-drivers=no
      make
      make install
      ```

Note

By default, the files are installed to /usr/local. You can specify a different location by altering the prefix option:

```
./configure --prefix=<new_location> --enable-gui=no --enable-drivers=no
```

where **<new_location>** is the location where you want to install the unixODBC libraries.

e) If you will be dynamically loading the driver manager from /prefix/lib, extract libodbc.a, libodbcinst.a, and libodbccr.a.

```
ar -x -X 64 libodbc.a
ar -x -X 64 libodbcinst.a
ar -x -X 64 libodbccr.a
```

f) Create the dynamically linked library.

```
ln -s libodbcinst.so.1 libodbcinst.so
```

3. Add the unixODBC library to the $LIBPATH environment variable.
For example:

```
export LIBPATH=<install_path>:$LIBPATH
```

where **<install_path>** is the location where all the unixODBC libraries are installed.

For the latest supported versions, refer to the *Product Availability Matrix* available at http://service.sap.com/PAM.

1.6.5.1.3.3 To build and install the unixODBC driver for Solaris

To install the unixODBC driver, you must be using a version of Solaris supported by SAP Data Services, have the Sun C++ compiler (version 5.5 or greater), and download the unixODBC source.

1. Download and extract the unixODBC package.

a) Download unixODBC-2.2.12.tar.gz from http://www.unixodbc.org to the $TEMP directory.

Note

The downloaded file will be named unixODBC-2.2.12.tar.tar.

b) Rename the unixODBC package.

```
mv unixODBC-2.2.12.tar.tar unixODBC-2.2.12.tar.gz
```

c) Extract the package with gunzip and tar.

```
gunzip unixODBC-2.2.12.tar.gz
tar -xvf unixODBC-2.2.12.tar
```

d) Change to the newly created `unixODBC-2.2.12` directory.

```
cd $TEMP/unixODBC-2.2.12
```

2. Make the libraries and programs.

a) Ensure that CC (the C++ compiler) is in the `PATH` environment variable, and add it if necessary.

○ Using ksh:

```
export PATH=/home4/thirdparty/software/sunonecc/8.0-sj/SUNWspro/bin/CC:
$PATH
```

○ Using csh:

```
setenv PATH /home4/thirdparty/software/sunonecc/8.0-sj/SUNWspro/bin/CC:
$PATH
```

b) Build the package using the standard GNU autoconf process.

```
./configure CFLAGS="-xarch=v9" LDFLAGS="-xarch=v9" CXXFLAGS="-xarch=v9" --
enable-gui=no
make
make install
```

> **Note**
>
> By default, the files are installed to `/usr/local`. You can specify a different location by altering the prefix option:
>
> ```
> ./configure --prefix=<new_location>/unixODBC CFLAGS="-xarch=v9" LDFLAGS="-
> xarch=v9" CXXFLAGS="-xarch=v9" --enable-gui=no
> ```
>
> where **`<new_location>`** is the location where you want to install the unixODBC libraries.

3. Add the unixODBC library to the `$LD_LIBRARY_PATH` environment variable.
For example:

```
export LD_LIBRARY_PATH=<install_path>/unixODBC/lib:$LD_LIBRARY_PATH
```

where **`<install_path>`** is the location where all the unixODBC libraries are installed.

For the latest supported versions, refer to the *Product Availability Matrix* available at *http://service.sap.com/PAM* .

1.6.5.1.4 Troubleshooting

You might need to troubleshoot the following situations:

- To determine whether all dependent libraries are set properly in the environment variables, you can use the `ldd` command on the ODBC driver manager library and the ODBC driver library.
For example:

```
ldd tdata.so
```

If you see that any dependent libraries are missing, ensure that you have added the environment settings to the session running the job service, or consult your ODBC driver vendor's documentation.

- If an error occurs when using the Connection Manager, invoke it from the command line by using the `-c` option, and use the `-d` option to show details in the log.
  For example:

  ```
  $LINK_DIR/bin/DSConnectionManager.sh -c -d
  ```

  The log file path is `$LINK_DIR/log/DSConnectionManager.log`.
  Possible errors include the following:
  - The Connection Manager cannot connect to database
  - The Data Services Job Server cannot connect to database

1.6.5.2 Configuring ODBC drivers on UNIX for data sources using DSN-less connections

On UNIX and Linux platforms, SAP Data Services requires an ODBC driver library for ODBC data sources using DSN-less connections. The UNIX ODBC driver configuration file:

- Contains driver names that reference ODBC driver libraries.
- Is an ODBC instance file referenced by the `$ODBCINST` environment variable.

> ➜ Tip
>
> For natively supported ODBC databases, you do not need to manually configure the ODBC instance file. Instead, set `$ODBCINST` to the name of the ODBC instance file and use the Data Services Connection Manager to properly configure the ODBC driver library.

1.6.5.2.1 Configuring native ODBC drivers on UNIX

Run the Connection Manager to configure the ODBC driver library and associated environment variables required to run a Data Services job that contains one of the following source or target database types using DSN-less connections:

- DB2 UDB
- Informix
- MySQL
- Netezza
- Oracle
- SAP HANA
- SAP Sybase IQ
- Teradata

> i Note
>
> For the most current list of database types supported for DSN-less connections, see the *Release Notes*.

1. For a DSN-less connection, set `$ODBCINST` to a file in which the Connection Manager will define the ODBC driver according to your input on the *Drivers* tab. Ensure that the file is readable and writable.
   For example:

   ```
   export ODBCINST=<dir-path>/odbc.inst
   touch $ODBCINST
   ```

2. Invoke the Connection Manager by entering the following command:

   `$LINK_DIR/bin/DSConnectionManager.sh`

3. Click the *Drivers* tab, and click *Add*.
   a) On the *Select Database Type* window, select the database type and click *OK*.
   The *Configuration for...* window appears with the value filled in for *ODBC Inst File*.
   b) Provide values for the driver properties. For the relevant driver properties for each database type, see *Properties for ODBC data sources using DSN-less connections* [page 70].
4. Provide values for the following properties (they will not be saved for further use).
   - Server name
   - Port (if relevant)
   - Database (if relevant)
   - User name
   - Password
5. If you want to test the connection, click *Test Connection*.
6. Click *Restart Services* to restart the following services:
   - Both the EIM Adaptive Process Service and Data Services Job Service if Data Services is installed at the same location (machine and folder) as Information Platform Services (IPS) or BI platform. A prompt will appear for the CMS password.
   - Only the Data Services Job Service if Data Services is installed without IPS or BI platform.
7. If you will run another command such as the Repository Manager, source the `al_env.sh` script to set the environment variables.

   By default, the script is located at `<LINK_DIR>/bin/al_env.sh`.

1.6.5.2.2 Properties for ODBC data sources using DSN-less connections

The Connection Manager configures the `$ODBCINST` file based on the property values that you enter on the *Drivers* tab. The following table lists the properties that are relevant for each database type.

Database Type	Properties on Drivers tab
MySQL	• ODBC Inst File • Driver Version • Unix ODBC Lib Path • Driver Name • Driver • Server Name

Database Type	Properties on Drivers tab
	• Port • Database • User Name • Password
SAP HANA	• ODBC Inst File • Driver Version • Driver Name • Driver • Server Name • Port • User Name • Password
Teradata	• ODBC Inst File • Driver Version • Driver Name • Driver • Server Name • User Name • Password
Netezza	• ODBC Inst File • Driver Version • Driver Name • Driver • Server Name • Port • Database • User Name • Password
SAP Sybase IQ	• ODBC Inst File • Driver Version • Driver Name • Driver • Server Name This is also known as the host name. For example, a host name may be: vanpgc13b9 • Port • Engine Name This is also known as the server name. For example, a server name may be: vanpgc13b9_iqdemo • Database

Database Type	Properties on Drivers tab
	• User Name • Password
Informix	• ODBC Inst File • Driver Version • Driver Name • Informix Home Path • Server Name • Database • User Name • Password
DB2 UDB	• DB2 Client Path • Driver Version • Server Name • Port • Database • User Name • Password
Oracle	• Oracle Home Path • Driver Version • Server Name • Port • SID • User Name • Password

1.6.6 Configuring other ODBC data sources

In addition to the natively-supported ODBC data sources, Data Services can access other ODBC data sources when you use the bundled DataDirect ODBC driver or another ODBC driver.

Related Information

Configuring native ODBC data sources [page 61]

1.6.6.1 To configure DataDirect ODBC

➡ Tip

It is recommended that you use the Data Services Connection Manager to configure ODBC data sources such as Microsoft SQL server and DB2 on zSeries or iSeries. The Connection Manager is an interactive user interface that simplifies the manual configuration steps of the DataDirect ODBC driver. For details about using the Connection Manager, see *Configuring native ODBC data sources* [page 61].

If you want to use the DataDirect ODBC driver to connect to ODBC data sources such as Microsoft SQL server and DB2 on zSeries or iSeries from Data Services on a Linux or Unix platform, follow these steps:

1. Add the data source to the Data Services ODBC driver manager configuration file (`<LINK_DIR>/bin/ds_odbc.ini`).
   For Microsoft SQL Server:

   ```
   [test_Microsoft_SQL_SERVER]
   Driver = <install_location>/lib/libodbc.so
   RebrandedLib = TRUE
   ```

   where `<install_location>` is the location of the DataDirect ODBC driver.

   For DB2 on zSeries or iSeries:

   ```
   [test_DB2]
   Driver = <install_location>/lib/libodbc.so
   RebrandedLib = TRUE
   ```

   where `<install_location>` is the location of the DataDirect ODBC driver.

   i Note

   `RebrandedLib = TRUE` is required when using the SAP rebranded Data Direct driver.

2. Add the data source to the ODBC vendor's configuration file (referenced by `$ODBCINI`).

   i Note

   The version number and driver filenames are subject to change with each release. Access `$LINK_DIR/DataDirect/odbc/odbc.ini` to view the current version information.

   i Note

   `EnableQuotedIdentifiers = 1` is required for Microsoft SQL server

   For Microsoft SQL Server:

   ```
   [test_Microsoft_SQL_SERVER]
   Driver=<install_location>/lib/[DA][DD]msss<xx>.so
   Description=DataDirect <current version number> SQL Server Wire Protocol
   AlternateServers=
   AnsiNPW=Yes
   ConnectionRetryCount=0
   ConnectionRetryDelay=3
   Database=<database_name>
   ```

```
EnableQuotedIdentifiers=1
HostName=<SQL_Server_host>
LoadBalancing=0
LogonID=
Password=
PortNumber=<SQL_Server_server_port>
QuotedId=No
ReportCodePageConversionErrors=0
DriverExpirationBehavior=1
```

where **`<install_location>`** is the location of the DataDirect ODBC driver.

For DB2 on zSeries or iSeries:

```
[test_DB2]
Driver=<install_location>/lib/[DD][DA]db2<xx>.so
Description=DataDirect <current version number> DB2 Wire Protocol
AddStringToCreateTable=
AlternateID=
AlternateServers=
ApplicationUsingThreads=1
AuthenticationMethod=0
CatalogSchema=
CharsetFor65535=0
#Collection applies to z/OS and iSeries only
Collection=<collection_name>
ConnectionRetryCount=0
ConnectionRetryDelay=3
#Database applies to DB2 UDB only
Database=<database_name>
DefaultIsolationLevel=1
DynamicSections=200
EncryptionMethod=0
GrantAuthid=PUBLIC
GrantExecute=1
GSSClient=native
HostNameInCertificate=
IpAddress=<DB2_server_host>
LoadBalancing=0
#Location applies to z/OS and iSeries only
Location=<location_name>
LogonID=
Password=
PackageCollection=<package_collection>
PackageOwner=
ReportCodePageConversionErrors=0
TcpPort=<port number>
TrustStore=
TrustStorePassword=
UseCurrentSchema=1
ValidateServerCertificate=1
WithHold=1
XMLDescribeType=-10
```

where **`<install_location>`** is the location of the DataDirect ODBC driver.

3. Run the `$LINK_DIR/DataDirect/odbc/odbc.sh` script to add the environment settings to the session running the job service.

1.6.6.2 Driver manager configuration file for DSN connections

Enclose data source names in square brackets. Properties follow on subsequent lines and use PropertyName = PropertyValue. For example:

```
[test_source]
Driver     = /path/to/driver
OdbcConformanceLevel     =
LazyLoading =
ODBC64SqlHandleSize =
ODBC64SqlLenSize     =
DriverUnicodeType     =
```

In this example, test_source is the name of data source that can be loaded using the specified driver library file. Default values apply when optional properties are left blank.

Follow these guidelines when editing the **<LINK_DIR>**/bin/ds_odbc.ini file:

- Each data source name must at least have a driver property defined, which allows the driver manager to load the driver when connecting to the database.
- The pound sign (#) as the first character in any line denotes a comment.
- All leading blanks and trailing blanks in data source names and properties are ignored.

The following table lists the data source configuration parameters for ds_odbc.ini (and ds_odbc.ini.sample):

Key	Required	Valid value	Example
Driver	Yes	A full path including the ODBC driver library name. The directory containing the dependent libraries must be in the shared library path (for AIX, LIBPATH; for Solaris or Linux, LD_LIBRARY_PATH). Check vendor documentation for what you need to add to the shared library path.	Driver=/home/mysql/myodbc/lib/libmyodbc3_r.so

OdbcConformanceLevel	No	A decimal value specifying the ODBC conformance level of driver. Default value is 0, in which case the driver detects by loading 2.x followed by 3.x functions from the driver. When any value greater than or equal to 4.0 is specified, the driver manager prints a run time error.	OdbcConformanceLevel =0
			OdbcConformanceLevel =3.0

Key	Required	Valid value	Example
		Note An ODBC driver can be compliant to either 2.x or 3.x or both. The UNIX ODBC driver manager detects if the driver is 2.x or 3.x compliant and loads the respective compatible ODBC API functions. If the driver is both 2.x and 3.x compliant, then the driver manager only loads the 2.x ODBC API. You can override this behavior by specifying, for example, `OdbcConformanceLevel = 3.0`. As a result, the ODBC driver manager only loads 3.x ODBC API functions.	
`LazyLoading`	No	You can specify a Boolean `TRUE`/`YES` or `FALSE`/`NO`. Default value is `FALSE`.The UNIX ODBC Driver Manager loads the ODBC driver and instructs the operating system to load all of its dependent libraries. This flag is useful when certain dependent libraries of the ODBC driver are not required and the ODBC vendor recommends to load the library in lazy mode.	`LazyLoading=TRUE`
`ODBC64SqlHandleSize`	Yes	`32` or `64` If blank or other, the software uses the default value of `64`. The standard definition of the SQLHANDLE data type in 64-bit ODBC is 64-bit integer. However, some ODBC drivers do not conform to this standard; therefore, use this parameter to specify the actual size of SQLHANDLE. DataDirect 64-bit ODBC drivers conform to the	`ODBC64SqlHandleSize=64`

Key	Required	Valid value	Example
		standard, so ignore or set to 64 for DataDirect. For other 64-bit ODBC drivers, contact your vendor to determine the actual size of SQLHANDLE. i Note This option is required only for 64-bit platforms.	
`ODBC64SqlLenSize`	Yes	`32` or `64` If blank or other, the software uses the default value of `64`. The standard definition of the SQLLEN data type in 64-bit ODBC is 64-bit integer. However, some ODBC drivers do not conform to this standard; therefore, use this parameter to specify the actual size of SQLLEN. DataDirect 64-bit ODBC drivers conform to the standard, so ignore or set to 64 for DataDirect. For other 64-bit ODBC drivers, contact your vendor to determine the actual size of SQLLEN. i Note This option is required only for 64-bit platforms.	`ODBC64SqlLenSize=64`
`DriverUnicodeType`	Yes	`1` (for UTF16) `2` (for UTF8) If blank, other, or not detectable, the software uses the default value of `2`. This integer value specifies the ODBC driver Unicode type. DataDirect SQL Server ODBC driver only supports W functions; for this driver, specify `2`.	`DriverUnicodeType=2`

Key	Required	Valid value	Example
		ℹ Note This option is required only for ODBC drivers that only support W functions.	

1.6.6.3 To configure Neoview ODBC

To use the Neoview Transporter on UNIX, you must also install the following software components:

- Neoview Transporter Java Client
- Java JRE version 1.5 or newer
- Neoview JDBC Type 4 driver
- Neoview ODBC UNIX drivers
- Neoview Command Interface

1. Run the `dsdb_setup.sh` script to set the Data Services ODBC configuration and associated environment variables required to run a Data Services job that contains a Neoview source or target.

   You need to provide the following information when you run the script:

   - The absolute location of the `odbc.ini` file in which the Neoview DSN is defined
   - The database version
   - The location of the database client

   By default, the script is located at **`<LINK_DIR>`**`/bin/dsdb_setup.sh`.
2. Add the data source to the ODBC vendor's configuration file (referenced by `$MXODSN`). For example:

```
[test_neoview]
Driver = <install_location>/libhpodbc_drvr[64].so
Description = Default Data Source
Catalog = NEO
Schema = <schema_name>
DataLang = 0
FetchBufferSize = SYSTEM_DEFAULT
Server = TCP:<ip_address>:<port_number>
SQL_ATTR_CONNECTION_TIMEOUT = SYSTEM_DEFAULT
SQL_LOGIN_TIMEOUT = SYSTEM_DEFAULT
SQL_QUERY_TIMEOUT = NO_TIMEOUT
ServiceName = HP_DEFAULT_SERVICE
```

   where **`<install_location>`** is the location of your HP Neoview installation.
3. Run the `al_env.sh` script to set the environment variables.

   By default, the script is located at **`<LINK_DIR>`**`/bin/al_env.sh`.
4. Use the Server Manager to restart the Data Services job service.
5. Stop and restart the Central Management Server (CMS) and its services to refresh the ODBC environment.

   a) Navigate to **`<BIP_INSTALL_DIR>`**.

b) Stop the CMS and its services:

```
./stopservers
```

c) Restart the CMS and its services:

```
./startservers
```

> **Note**
>
> You must also change the regional settings to UTF-8 to process multi-byte data.

1.6.7 Using the ODBC Driver Selector on Windows for server name connections

Run the Data Services ODBC Driver Selector on Windows to configure the ODBC driver library required for a database using server name (also known as DSN-less) connections for the Data Services repository or as a source or target database in a Data Services job.

> **Note**
>
> For the most current list of database types and versions supported for DSN-less or TNS-less connections, see the Supported Platforms (Product Availability Matrix) *https://service.sap.com/PAM*.

1. Invoke the ODBC Driver Selector by opening a Command Prompt window and entering the following command:

   ```
   %LINK_DIR%/bin/ODBCDriversSelector.exe
   ```

2. Go to the database type and version in the column *Database version*, and click the cell under the column *ODBC Drivers* to display a list of existing drivers that Data Services detected and the current state of the driver.

   > **Note**
   >
   > The list of drivers in the ODBC Driver Selector is the same as the list in the Windows ODBC Data Source Administsrator for data sources using DSN connections. The state in the ODBC Driver Selector will have a state of “Installed” for these drivers. However, if you uninstall a driver, the ODBC Driver Selector state is “Not Installed”.

3. Select the ODBC Driver for your database type and click *OK*.

1.6.8 Using the Repository Manager

Use the Repository Manager to check the version, to upgrade, or to create a repository after installation.

⚠ Caution

It's recommended that you do not use database tools to attempt to quickly replicate additional repositories. By using the Repository Manager to create and seed multiple repositories individually, you can avoid potential issues related to the configuration of repository objects.

1. On Windows, choose *Start* > *Programs* > *SAP Data Services 4.2* > *Data Services Repository Manager* to open the Repository Manager.

   You can also access the Repository Manager from the command line on both Windows and UNIX platforms. For more information, see the command-line reference appendix.
2. If you are creating a new repository, ensure that you created a database for the new repository to use.
3. In the Repository Manager window, select the database type and version of your repository.
4. For a DB2, MySQL or SAP HANA database, the default connection type is DSN-less (the *Use data source name (DSN)* checkbox is not selected).
   - If you want to use a DSN-less connection, type in the *Database server name*, *Database name* (for DB2 and MySQL) and *Port*.
   - If you want to use a DSN connection, select the check box for *Use data source name (DSN)* and enter the *Data source name*.
5. For an Oracle database, the default connection type is TNS-less (the *Use TNS name* checkbox is not selected).
   - If you want to use a TNS-less connection, type in the *Hostname*, *SID*, and *Port*
   - If you want to use a TNS connection, select the check box for *Use TNS name* and enter the *TNS name*.
6. For other database types, complete the information.
7. Enter the user name and password that you want to use for your repository.
8. Select one of the following repository types:

Repository type	Description
Local	(Default) Stores definitions of objects in your local repository.
Central	Stores definitions of objects in a central repository for multiple-user users.
Profiler	Stores information generated by the Data Profiler for determining the quality of your data.

9. If you are creating a new repository, click *Create*. If you are upgrading an existing repository, click *Upgrade*.
10. If you want to create or upgrade another repository, repeat steps 1 through 6.
11. When you finish creating or upgrading repositories, click *Close*.

ℹ Note

Before you can access the repository, you must associate it with a Job Server and register it in the Central Management Console (CMC)

Related Information

DSN-less and TNS-less connections [page 30]
To register a repository in the CMC [page 52]
Using the Server Manager on Windows [page 84]

1.6.9 Using the License Manager

License Manager lets you manage your product activation keycodes—the alphanumeric codes that are referred to each time that you run certain software. By using License Manager, you can view, add, and remove product activation keycodes for SAP solution portfolio software (such as SAP Data Services) that require them.

> **Note**
>
> License Manager accesses keycodes on the local system only; you cannot access the keycodes from a remote system. When updating keycodes, make the changes on all SAP Data Services computers by launching License Manager on each computer, including Designer and Job Server computers.

> **Note**
>
> If you are running a Windows operating system, you will not be able to add or remove license keycodes unless you have Administrator privileges. For those with non-administrator privileges, the License Manager interface will appear in read-only mode. For the command-line interface, only the `-v` and `--view` parameters are available for use.

1.6.9.1 To configure License Manager on Unix

Before you can use License Manager on UNIX platforms, you need to set the environment variable `BOE_REGISTRYHOME`. If you've already configured the SAP Data Services environment by running `al_env.sh` script, the `BOE_REGISTRYHOME` variable should already be set. If the variable has not been set, manually add it to your `.profile`, `.login`, or `.cshrc` file.

If you use Bourne shell, add product entries to your `.profile` or `.login` file.

```
BOE_REGISTRYHOME=$LINK_DIR/registry ; export BOE_REGISTRYHOME
```

If you use C shell (Berkeley), add product entries to your .cshrc file.

```
setenv BOE_REGISTRYHOME $LINK_DIR/registry
```

1.6.9.2 To start License Manager

You can run License Manager after the SAP Data Services installation has completed.

On Windows

Choose *Start* > *Programs* > *SAP Data Services 4.2* > *License Manager*.

> **Note**
>
> You can also use License Manager in command-line mode.

On UNIX

Run `LicenseManager` from the command line without specifying any options:

```
$ cd $LINK_DIR/bin
$ ./LicenseManager
```

> **Note**
>
> If X-Windows is not available, you can use License Manager in command-line mode.

1.6.9.3 To view product activation keycodes

1. Start License Manager.
   The License Manager window displays your keycode(s) sorted alphabetically.
2. Select a licensed product or feature in the *Registered Keycodes* tree to view detailed information:
   - Product or feature keycode
   - Whether the keycode is a trial version
   - Whether the keycode is expired
   - Number of days remaining until the keycode expires

Related Information

To start License Manager [page 82]

1.6.9.4 To add product activation keycodes

1. Start License Manager.
2. In the *Product Activation Keycodes* text box, enter the keycode(s) that you want to add (each keycode must be on a separate line) and click *Add*.
   The keycodes that will be added are displayed in the *Registered Keycodes* tree and highlighted.
3. When you are satisfied with the changes that will be made, click *Save*.
   The keycode highlighting is removed.

   > ➜ Tip
   >
   > If you do not want to save the keycode changes, close License Manager without saving the changes.
4. After you have saved your changes, click *Close* to exit License Manager.
5. To make sure the new keycode(s) take effect, restart the software.

Related Information

To start License Manager [page 82]

1.6.9.5 To remove product activation keycodes

1. Start License Manager.
2. In the *Registered Keycodes* tree, select the keycode(s) that you want to remove and click *Remove*.
   The keycodes that will be removed are crossed out, and any affected nodes are highlighted.
3. When you are satisfied with the changes that will be made, click *Save*.
   The crossed-out keycodes are removed.

   > ➜ Tip
   >
   > If you do not want to save the keycode changes, close License Manager without saving the changes.
4. After you have saved your changes, click *Close* to exit License Manager.
5. Restart the software.

Related Information

To start License Manager [page 82]

1.6.10 Using the Server Manager on Windows

Use the Server Manager to create, edit, or delete Job Servers and Access Servers after installation.

1. Choose *Start* > *Programs* > *SAP Data Services 4.2* > *Data Services Server Manager*.
   The Server Manager utility window opens. This window shows the Job Servers and Access Servers currently configured to run on your computer.
2. Configure Job and Access servers as needed.
3. In the Server Manager window, click *Restart*.

1.6.10.1 To configure Job Servers

1. Open the Server Manager, click the Job Server tab and click *Edit*.
2. Decide which configuration task to perform:
   - To add a new Job Server, click *Add*.
     Continue to the remaining configuration steps.
   - To edit an existing Job Server, select the Job Server and click *Edit*.
     Continue to the remaining configuration steps.
   - To remove an existing Job Server, select the Job Server and click *Delete*.
     No additional configuration steps are required.

   > Note
   >
   > If the Job Server has associated repositories, you must first delete those and then click *OK* before you can delete the Job Server.

3. In the Job Server Properties window, enter configuration information for the Job Server.
4. In the Associated Repositories section, configure any local or profiler repositories that you want to associate with the Job Server. Each Job Server must be associated with at least one local repository.
   a) If you want to use a DSN-less connection (for DB2, MySQL, SQL Anywhere, or SAP HANA database types), clear the *Use data source name (DSN)* checkbox.
   b) If you want to use a TNS-less connection for an Oracle database type, clear the *Use TNS name* checkbox.
   c) When you have finished configuring associated repositories, including one default, click *OK*.
5. Click *OK* to return to the Server Manager window.
6. Click *Restart* to restart the services with any updated configurations.

1.6.10.1.1 Job Server properties

Property	Description
Job Server name	Specifies a name that uniquely identifies the Job Server.

Property	Description
Job Server port	Specifies the TCP/IP port that the Job Server uses to receive commands from the Designer and the Access Server. If a computer hosts multiple Job Servers, each Job Server must have a unique port number. Additionally, the post number must not be used by another process on the computer. If you are unsure of which port number to use, use the default port number and increment it for each additional Job Server that you configure.
Support adapter and message broker communication	Enables communication between the Job Server and adapters. Each computer that hosts adapters must have exactly one Job Server designated to manage them.
Use SSL protocol for adapter, message broke and communication	Enables SSL security on the communication paths between the Job Server and any adapters or message brokers.
Communication port	Specifies the TCP/IP port number that the Job Server uses for communicating with adapters. The default port is 4001.

1.6.10.1.2 To configure associated repositories

Each Job Server must be associated with at least one local repository, and can be associated with other local and profiler repositories. Configure associated repositories in the Associated Repositories section of the Job Server Properties window in the Server Manager.

To add an associated repository

1. Click *Add* to associate a new local or profiler repository with the Job Server.
2. Enter the required connection information for your repository database. The details required vary depending on the database type.
3. Enter the user name and password that you want to use for your repository.
4. Check *Default repository* if this is the default repository for the Job Server. You must specify exactly one default repository.

   > Note
   >
   > Do not check *Default repository* if you are adding a profiler repository.

5. Click *Apply* to save your entries and associate the repository with the Job Server.

The associated repository entry updates with the Job Server's computer name and port number.

To edit an associated repository

1. Select the repository you want to change and click *Edit*.
2. Under Repository Information, enter the password.
3. Check or uncheck *Default repository*, indicating whether this is the default repository for the Job Server.
4. Click *Apply* to save the changes to the Job Server configuration.

> Note
>
> You can change only whether an associated repository is the default for the Job Server. If you need to make other changes, delete the existing associated repository and add a new one with the updated configuration information.

To delete an associated repository

1. Select the repository you want to delete and click *Delete*.
2. Under Repository Information, enter the password.
3. Click *Apply* to remove the associated repository from the Job Server configuration.

1.6.10.1.3 To resynchronize associated repositories

Situations when you must resynchronize the Job Server and the local repository include:

- The Job Server information is not available or not correct in the local repository.
- You have uninstalled Data Services and are reinstalling the same version without creating a new local repository.
- You created a new local repository using the Repository Manager after creating a repository and Job Server when you installed Data Services.

To resynchronize Job Servers:

1. In the Job Server Configuration Editor window, select the name of your Job Server.
2. Click *Resync with Repository*.
3. In the Job Server Properties window, select an associated local repository.
4. Click *Resync*.
5. When asked whether to update this associated repository with this local machine information, click *OK*.
6. Under *Repository Information*, enter the local repository password.
7. Click *Apply*.
8. Click *OK* on the Job Server Properties window.

1.6.10.2 To configure run-time resources

1. In the Server Manager window, click the Run-time resources tab.
2. For the *Specify a directory with enough disk space for pageable cache* option, accept the default directory (**<LINK_DIR>**`\Log\PCache`) or click the ellipses button to browse to a different directory.

   > i Note
   >
   > For memory-intensive operations such as Group By, Order By, and Detailed profiling, specify a pageable cache directory that fulfills the following criteria:

   - The directory contains enough disk space for your data. To estimate the amount of space required for pageable cache, consider factors such as:
     - Number of concurrently running jobs or data flows.
     - Amount of pageable cache required for each concurrent data flow.
   - The directory exists on a separate disk or file system from the SAP Data Services system and operating system (such as the `C:` drive on Windows, or the root file system on UNIX systems).
   - The directory limits the disk space that data flows consume. The pageable cache uses all available disk space on the file system that contains the pageable cache directory. So, to limit the disk space that data flows consume, create a file system (or partition on Windows) with a limited size. Use the new file system (partition on Windows) as the pageable cache directory.
     The software uses this directory in the following situations:
     - For pageable caching, which is the default cache type for data flows.
     - When selecting a file transfer type and Automatic is specified in the Data_Transfer transform.
3. In the *Peer-to-peer options* area, change the values for *Start port* and *End port* to restrict the number of ports used by the software. The default values for *Start port* and *End port* are `1025` and `32767`, respectively.

   The software uses these ports for peer-to-peer communications when sending data between data flows or sub data flows.

   > i Note
   >
   > If you want to enable SSL security on the communication paths between data flows and sub data flows, select *Use SSL protocol*.

4. Click *Apply* to save any configuration changes.

Related Information

Performance Optimization Guide: Caching data [page 2129]

Reference Guide: Data_Transfer [page 1071]

Performance Optimization Guide: Using grid computing to distribute data flows execution [page 2160]

1.6.10.3 To configure Access Servers

When you configure the location for an Access Server installation, SAP Data Services creates space for the Access Server log files.

1. Open the Server Manager, click the Access Server tab and click *Edit*.
2. Decide which configuration task to perform:
   - To add a new Access Server, click *Add*.
     Continue to the remaining configuration steps.
   - To edit an existing Access Server, select the Access Server and click *Edit*.
     Continue to the remaining configuration steps.
   - To remove an existing Access Server, select the Access Server and click *Delete*.
     No additional configuration steps are required.
3. In the *Access Server Properties* window, enter the Access Server configuration information and click *OK*.

Property	Description
Directory	Specifies the location of the log files for this instance of the Access Server. Click the ellipses button to browse to the Log directory under the directory where you installed the software. Do not change this value after the initial configuration.
Communication Port	Specifies the port on this computer that the Access Server uses to listen for incoming messages from clients. Make sure that this port number is unused and is unique for each Access Server.
Parameters	Specify any additional Access Server parameters. Note Additional Access Server parameters can be viewed by typing `AL_AccessServer` at the command line. For more information, see “Real Time Performance” in the *Management Console Guide*.
Use SSL protocol	Enables SSL security for real-time messaging on this Access Server.
Enable Access Server	Controls whether the Access Server is automatically started when the Data Services service starts.

4. Click *OK* to return to the Server Manager window.
5. Click *Restart* to restart the services with the updated configuration.

1.6.10.4 To configure SSL paths

Use the Server Manager to configure the paths to SSL certificates and keyfiles.

Note

By default, the paths for the SSL certificate and keyfiles are automatically configured during installation. You do not need to change them unless you want to use your own certificates.

Note

If you change the SSL certificate configuration, you must resync all repositories associated with the Job Server before you can run jobs successfully.

1. Open the Server Manager and click the SSL tab.
2. Specify the locations of the server certificate file, the server private key file, and the trusted certificates folder.

   Note

   The server certificate must be in PEM format. Valid extensions for certificates in the trusted certificates folder include `.pem`, `.crt`, and `.cer`. Regardless of the file extension, all certificate file contents must be in PEM format.
3. If you want to specify a private key password file, select *Use server private key password file* and specify the location of the password file.
4. Click *Close and Restart* to close the Server Manager and restart any Data Services servers on the machine with the updated certificate information.

Note

The certificate information specified in the Server Manager applies to all Data Services servers running on that physical machine. For example, any Job Servers, Access Servers, and so on.

1.6.10.5 Verifying that Job and Access servers are running

To verify that Job Servers are running:

1. Check in the Windows Task Manager *Processes* tab for:
   - `al_jobservice.exe` (represents the SAP Data Services service)
   - `al_jobserver.exe` (one per Job Server)
   - `AL_AccessServer.exe` (one per Access Server)
2. If you do not see all the processes expected, check for error messages in the Job Server event log in **`<LINK_DIR>`**`/log/`**`<JobServer name>`**`/server_eventlog.txt`.

   Note

   Access Server logs are in **`<AccessServerPathName>`**`/error_mm_dd_yyyy.log`

1.6.11 Using the Server Manager on UNIX systems

Use the Server Manager to create, edit, or delete Job Servers and Access Servers after installation.

The Server Manager displays the following:

Job Server information

Option	Description
Server name	This name uniquely identifies the Job Server. The Job Server name cannot be changed.
TCP/IP port number	The port number is a TCP/IP port that the Job Server uses to receive commands from the Designer and an Access Server. If a computer hosts multiple Job Servers, each Job Server must have a unique port number. Choose a port number that is not used by another process on the computer. It's recommended that you use 3500. If you are unsure of which port number to use, use the default port number and increment it for each additional Job Server you configure.
Supports adapter communication on port	If this computer hosts adapters, you must designate one (and only one) Job Server to support them. Once a Job Server is set to support adapters (a port is entered and saved), it is marked on the Job Server Configuration screen with this label.

Run-time resource information

Option	Description
Pageable cache directory	This directory contains the pageable cache that the software uses for memory-intensive operations and for file transfer types when *Automatic* is specified in the Data_Transfer transform.
Start port	The software uses this starting port number for peer-to-peer communication between data flows or sub data flows that are running on different Job Servers. The default is 1025.

Access Server information

Option	Description
Server number	This sequence number uniquely identifies the Access Server on this machine. The Access Server number cannot be changed.
Directory	The directory containing Access Server information.
Communication port	This port number is used to communicate between the Access Server and the Administrator. The default is 4000.
Parameters	Additional parameters used by the Access server.

Option	Description
	View Access Server parameters by typing **AL_AccessServer** at the command line. For more information, see "Real Time Performance" in the *Management Console Guide*.
Enable	Enter **Y** to activate the Access Server.

Job service information

Option	Description
Service executable path	The directory containing AL_JobService information.
Status	Status of the Data Services service: - Running - Not running

SMTP Server information

Option	Description
Server	The name or IP address of the SMTP server (for example, mail.company.com).
Sender	The email address that will appear in the *From* field of the email.

1.6.11.1 To configure Job Servers on UNIX

1. Ensure required environment variables are set, and run the Server Manager.

   ```
   $ cd $LINK_DIR/bin/
   $ . ./al_env.sh
   $ ./svrcfg
   ```

   The Server Manager main screen appears.
2. Enter **3** to configure a Job Server.
   The Job Server information screen appears.

   > Note
   >
   > The repository information for each configured Job Server is displayed in one of the following formats:
   >
   > ○ For a DSN or TNS connection:
   >
   > | Database Type | Format of Repository String |
   > |---|---|
   > | Oracle | **<username>**@**<TNSname_user>** |

Database Type	Format of Repository String
SAP HANA	`<username>@<DSNname_user>`
DB2	
MySQL	
SQL Anywhere	

- For a server name connection (also known as DSN-less or TNS-less connection):

Database Type	Format of Repository String
Oracle	`<username>@<server_SID_user>`
SAP HANA	`<username>@<server_port_user>`
MySQL	`<username>@<server_database_user>`
DB2	

- For SAP Sybase:
  `<username>@<server_database_user>`

3. Enter the command for the configuration task you want to perform:

Command	Configuration task
`c`	Add a new Job Server.
`e`	Edit an existing Job Server.
`d`	Delete an existing Job Server.
`a`	Add a repository connection to a Job Server.
`u`	Update a repository connection on a Job Server.
`r`	Remove a repository connection from a Job Server.
`s`	Set the default repository connection for a Job Server.
`y`	Resynchronize a Job Server configuration with a repository. You must resynchronize your Job Server and repository when: ○ You have uninstalled Data Services and are reinstalling the same version without creating a new repository. ○ You have created a new repository using the Repository Manager after installing the software. If you resynchronize your Job Server configuration with a repository, you must re-add a connection for this repository to the Administrator. For more information, see the *Management Console Guide*.

4. When you add or edit a Job Server, you must specify additional configuration details:
   a) Enter the name for the Job Server.
   b) Specify the TCP/IP port that the Job Server uses to receive commands from the Designer and the Access Server.

> i Note
>
> If a computer hosts multiple Job Servers, each Job Server must have a unique port number. Additionally, the post number must not be used by another process on the computer.

If you are unsure of which port number to use, use the default port number and increment it for each additional Job Server that you configure.

c) If you want to manage adapters with the Job Server, enter **Y**.

d) If you want to manage adapter communication with the Job Server, specify the TCP/IP port number to use.

e) If you want to enable SSL on the adapter management communication paths used by the Job Server, enter **Y**.

5. When you add or edit a repository connection, you must specify the database connection information.

a) If you want to use a DSN-less connection (for a DB2, MySQL, or SAP HANA database), enter **N** when the Server Manager asks you if you want to use an ODBC data source.

b) If you want to use a TNS-less connection for an Oracle database, enter **N** when the Server Manager asks you if you want to use a TNS name.

c) If you want to use a DSN or TNS connection, you must specify the following additional database connection information:

Database	Required information
Oracle	The TNSNAME specified in `tnsnames.ora`
MySQL	The DSN entry specified in `odbc.ini`
SAP HANA	The DSN entry specified in `odbc.ini`
DB2	The DB2 instance name
SQL Anywhere	The DSN entry specified in `odbc.ini`

d) If your database type is SAP Sybase, specify the Sybase server name specified in the Interfaces file.

Note

The Server Manager for UNIX systems does not prompt for the repository password except when creating a Job Server or adding a repository. To update the repository password in the **<DS_COMMON_DIR>**`/conf/DSConfig.txt` file, enter **u**. All options use the updated password from `DSConfig.txt` file.

6. When you are satisfied with your configuration changes, enter **q** and then **x** to exit the Server Manager.

Related Information

DSN-less and TNS-less connections [page 30]

1.6.11.2 To configure run-time resources

1. Ensure required environment variables are set, and run the Server Manager.

```
$ cd $LINK_DIR/bin/
$ . ./al_env.sh
$ ./svrcfg
```

The Server Manager main screen appears.

2. Enter `4` to configure run-time resources.
   The run-time resource information screen appears.
3. Enter `e` to edit the run-time resource configuration.
4. Accept the default *Pageable Cache Directory*, or specify a different location.

   ⚠ Restriction

   The Pageable Cache Directory path cannot exceed 70 characters.

   ℹ Note

   For memory-intensive operations such as Group By, Order By, and Detailed profiling, specify a pageable cache directory that fulfills the following criteria:

   - The directory contains enough disk space for your data. To estimate the amount of space required, consider factors such as the number of concurrently running jobs or data flows and the amount of pageable cache required by each concurrent data flow.
   - The directory exists on a separate disk or file system from the Data Services system and operating system.
   - The directory limits the disk space that data flows consume. The pageable cache uses all available disk space on the file system that contains the pageable cache directory. To limit the disk space that data flows consume, create a file system with a limited size. Use the new file system as the pageable cache directory.

   The software uses this directory in the following situations:

   - For pageable caching, the default cache type for data flows. For more information, see the *Performance Optimization Guide*.
   - When the software selects a file transfer type and *Automatic* is specified in the Data_Transfer transform.
5. Change the values for *Start port* and *End port* to restrict the number of ports used by the software for peer-to-peer communications. The default values are 1025 and 32767, respectively.

   The software uses these ports for peer-to-peer communications when sending data between data flows or sub data flows that are running on different Job Servers.
6. Specify whether you want to use the SSL security protocol on the communication paths between data flows and sub data flows.
7. Enter `q` and then `x` to exit the Server Manager.

Related Information

Performance Optimization Guide: Using grid computing to distribute data flow execution [page 2160]

1.6.11.3 To configure Access Servers

When you configure the location for an Access Server installation, SAP Data Services creates space for the Access Server log files.

1. Ensure required environment variables are set, and run the Server Manager.

   ```
   $ cd $LINK_DIR/bin/
   $ . ./al_env.sh
   $ ./svrcfg
   ```

   The Server Manager main screen appears.
2. Enter **4** to configure an Access Server.
   The Access Server information screen appears.
3. Enter the command for the configuration task you want to perform:

Command	Configuration task
c	Create a new Access Server.
e	Edit an existing Access Server.
d	Delete an existing Access Server.

4. When you create or edit an Access Server, specify additional configuration details:
   a) If you are editing an existing Access Server, enter the number of the Access Server shown in the Access Server configuration information screen.
   b) Specify the directory for the Access Server.
   c) Specify the TCP/IP port that the Access Server should use for communication.

   > Note
   >
   > You can configure more than one Access Server on the same computer, but each must have separate ports. If you enter a port number already in use, an error message appears.

   d) Specify any additional parameters for the Access Server.

   > Note
   >
   > Additional Access Server parameters can be viewed by typing `AL_AccessServer` at the command line. For more information, see "Real Time Performance" in the *Management Console Guide*.

   e) Specify whether you want to use the SSL security for real-time messaging on this Access Server.
   f) Specify whether you want to enable the Access Server.
5. When you delete an Access Server, specify the number of the Access Server to delete.

   > Note
   >
   > When you delete an Access Server, all Access Servers are stopped. When you exit the Server Manager, any remaining Access Servers restart.

6. When you are satisfied with your configuration changes, enter **q** and then **x** to exit the Server Manager.

1.6.11.4 To configure SSL paths

Use the Server Manager to configure the paths to SSL certificates and keyfiles.

> Note
>
> By default, the paths for the SSL certificate and keyfiles are automatically configured during installation. You do not need to change them unless you want to use your own certificates.

1. Ensure required environment variables are set, and run the Server Manager.

   ```
   $ cd $LINK_DIR/bin/
   $ . ./al_env.sh
   $ ./svrcfg
   ```

   The Server Manager main screen appears.
2. Enter `7` to configure SSL paths.
   The SSL configuration information screen appears.
3. Enter `e` to edit the SSL configuration.
4. Specify the SSL configuration information when prompted:
   a) The path to the server certificate file
   b) The path to the server private key file
   c) Whether you want to use a private key password file and the path to that file
   d) The directory where your trusted certificates are stored

   > Note
   >
   > The server certificate must be in PEM format. Valid extensions for certificates in the trusted certificates folder include `.pem`, `.crt`, and `.cer`. Regardless of the file extension, all certificate file contents must be in PEM format.
5. When you are satisfied with your configuration changes, enter `q` and then `x` to exit the Server Manager.

The certificate information specified in the Server Manager applies to all Data Services servers running on that physical machine (for example, any Job Servers, Access Servers, and so on.)

1.6.11.5 To start or stop the service

The SAP Data Services service (`AL_JobService`) is a daemon associated with `$LINK_DIR` that starts locally-configured Job Servers and Access Servers and then monitors them and attempts to restart them if they are not running.

After you exit the Server Manager, `AL_JobService` automatically retrieves any changes made to Job Servers or Access Servers. You do not need to restart `AL_JobService`.

1. Run the Server Manager.

   ```
   $ cd $LINK_DIR/bin/
   $ . ./al_env.sh
   $ ./svrcfg
   ```

> ℹ Note
>
> The second command sets required environment variables before `./svrcfg` starts the Server Manager.

The Server Manager main screen appears.

2. Enter **`1`** to control the service (Job service).
3. Start or stop the Job service.
   - Enter **`s`** to start the Job service.
   - Enter **`o`** to stop the Job service.
4. Enter **`q`** and then **`x`** to exit the Server Manager.

1.6.11.6 To configure SMTP email

The Server Manager can be used to specify SMTP server settings for the `smtp_to` email function. For more information, see "To define and enable the smtp_to function" in the *Reference Guide*.

1.6.12 Configuring Metadata Browsing Service and View Data Service

The installation process of Data Services configures the following services (under the server EIMAdaptiveProcessingServer) with default settings.

- Metadata Browsing Service
- View Data Service

These services are used by Information Steward to connect and view data in profiling sources. You might want to change the configuration settings to more effectively integrate Information Steward with your hardware, software, and network configurations.

1. Go to the *Servers* management area of the CMC.
2. Expand *Service Categories* in the tree panel and select *Enterprise Information Management Services*.
3. Double-click **`<computername.>`** *EIMAdaptiveProcessingServer* in the list in the right pane.
4. On the *Properties* window, find the applicable service for which you want to change settings.
5. After making desired changes in the service, click *Save* or *Save & Close*.

> ℹ Note
>
> Not all changes occur immediately. If a setting cannot change immediately, the *Properties* window displays both the current setting (in red text) and the updated setting. When you return to the *Servers* management area, the server will be marked as *Stale*. When you restart the server, it will use the updated settings from the *Properties* dialog box and the *Stale* flag is removed from the server.

Related Information

Metadata Browsing Service configuration parameters [page 98]
You can change the following properties of the Metadata Browsing Service.

View Data Services configuration parameters [page 99]
You can change the following properties of the View Data Service.

1.6.12.1 Metadata Browsing Service configuration parameters

You can change the following properties of the Metadata Browsing Service.

Server Configuration Parameter	Description	Possible Values
Service Name	Name of the service configuration.	Alphanumeric string with a maximum length of 64. The Service Name cannot contain any spaces. Default value: MetadataBrowsingService
Maximum Data Source Connections	Maximum number of data source connections that can be opened at any time under a service instance.	integer. Default value: 200
Retry attempts to launch Service Provider	Maximum number of attempts to launch a new service provider when there is contention to access a shared service provider.	Default value: 1
Stateful Connection Timeout (seconds)	Maximum duration which a stateful connection is open. Stateful connections include SAP Applications and SAP BW Source.	Default value: 1200
Stateless Connection Timeout (seconds)	Maximum duration which a stateless connection is open. Stateless connections include all relational database sources.	Default value: 1200
Recycle Threshold	Maximum number of requests that will be processed by a service before the Data Services backend engine is recycled to free memory that was allocated for metadata browsing.	Default value: 50000
Log Level	Level of logging of trace messages to the log file. Note If there is more than one instance of Metadata Browsing Service configured in the CMS, the same level of information is collected from all	Information Steward logs: ● None: Logging disabled. ● Info: Logging disabled. (same as None) ● Finer: All traces, requests, and responses.

Server Configuration Parameter	Description	Possible Values
	instances. The log level defined for the first running service is the level used.	Data Services logs: ● None: Logging disabled. ● Info: All traces. ● Finer: All traces, requests, and responses. Default level is Info.
Collect Connection Statistics	Enable or disable the collection of statistic information for each open connection.	Default is enabled.
Listener Port	Port number used to communicate with the Data Services backend engine. If you change the port number, you must restart the EIMAdaptiveProcessingServer for the change to take effect.	Four-digit port number that is not currently in use. Default value: 4010
JMX Connector Port	Port number used for the JMX Connector. If you change the port number, you must restart the EIMAdaptiveProcessingServer for the change to take effect.	Four-digit port number that is not currently in use. Default value: 4011

1.6.12.2 View Data Services configuration parameters

You can change the following properties of the View Data Service.

Server Configuration Parameter	Description	Possible Values
Service Name	Name of the service configuration.	Alphanumeric string with a maximum length of 64. The Service Name cannot contain any spaces. Default value: ViewData
Listener Port	Port number used to communicate with the Data Services backend engine. If you change the port number, you must restart the EIMAdaptiveProcessingServer for the change to take effect.	Four-digit integer. Default value: 4012
JMX Connector Port	Port number used for the JMX Connector. If you change the port number, you must restart the EIMAdaptiveProcessingServer for the change to take effect.	Four-digit integer. Default value: 4013

Server Configuration Parameter	Description	Possible Values
Batch Size (kilobytes)	Size of the data to be stored in a view data response.	Minimum value: 1000 Maximum value: 50000 Default value: 1000
Minimum Shared Service Providers	Minimum number of shared Data Services backend engines that need to be launched at the startup time of the service.	Default value: 1
Maximum Shared Service Providers	Maximum number of shared Data Services backend engines that can be launched during the time to service the view data requests.	Default value: 5
Maximum Dedicated Service Providers	Maximum number of dedicated Data Services backend engines that can be launched at any instant of time.	Default value: 10
Recycle Threshold	Maximum number of requests that will be processed by a service before the Data Services backend engine is recycled to free memory that was allocated for viewing data.	Any integer. Default value: 200
Number of attempts to launch service provider	Number of attempts to be made to try launching the Data Services backend engine instance.	Default value: 1
Maximum idle time for shared service provider (minutes)	Maximum number of minutes that a Data Services backend engine can remain without processing any requests. After this time is exceeded, the Data Services backend engine is shut down.	Default value: 120
Log Level	Level of logging of trace messages to the log file. Note If there is more than one instance of View Data Service configured in the CMS, the same level of information is collected from all instances. The log level defined for the first running service is the level used.	Information Steward logs: • None: Logging disabled. • Info: Logging disabled. (same as None) • Finer: All traces, requests, and responses. Data Services logs: • None: Logging disabled. • Info: All traces. • Finer: All traces, requests, and responses. Default level is Info.

1.6.13 Data Services CMC application settings

You can change the following settings of the Data Services Application on the CMC.

Data Services Application	Description
History Retention Period	Number of days to retain the job execution history. Default value: 30 • If you enter 0, then no job history is maintained. • If you enter a negative number, then job history is deleted.
Job Server Log Retention Period	Number of days to retain the Job Server log files. Default value: 30 • If you enter 0, then no job history is maintained. • If you enter a negative number, then job history is deleted.
Enable SSL communication for Metadata Browsing and View Data Services	Specifies whether or not to use SSL communications for Metadata Browsing Service and View Data Service of the EIM Adaptive Processing Server. Other SAP software products, such as SAP Information Steward, use the Metadata Browsing Service and View Data Service service to browse and import metadata and to view the data in connections.
Use Default SSL Settings	Specifies whether or not to use the default SSL keystore and certificates. Default value: No If you specify `No`, then you must enter values in *KeyStore File*, *KeyStore Password*, and *Key Password*.
KeyStore File	File name of the keystore that contains the key and all the certificates that are part of the certificate chain involved in signing the key. Default value: `DSJavaKeyStore.keystore`
KeyStore Password	Password to the keystore file.
Key Password	Password to the key inside the keystore file.
Encryption passphrase	Passphrase to use for encrypting passwords that are sent as part of requests to the Metadata Browsing Service and View Data Service. Other SAP software products, such as SAP HANA, use this *Encryption passphrase* to encrypt passwords when sending an open connection request. The backend engine will use this passphrase to decrypt the password and process the open connection request.

1.7 Monitoring

1.7.1 Monitoring jobs

Using the Administrator, you can monitor job execution of any batch job in a connected repository. You can monitor jobs that you run from the Administrator or from the Designer.

This section discusses how you can use the Administrator to view a batch job's overall status and statistics.

1.7.1.1 To view overall status of executed jobs

The *Batch Job Status* page lists each batch job execution. Use this list to view the overall status of each execution and to access more detailed statistics and log files.

1. Select ▶ *Batch* ▸ `<repository>` ◢.

   To view jobs in all repositories from this page, select ▶ *Batch* ▸ *All Repositories* ◢. (The All Repositories option appears under the Batch Job node if more than one repository is connected to the Administrator.)

   The *Batch Job Status* page shows each instance of job execution for the selected repository.
2. You can filter the list of batch jobs displayed by selecting a job name and/or when the job executed.

   To filter by job, select the job name from the drop-down *Job name* list. Or type the name, or type part of the name and a wildcard character (`%` or `*`), into the wildcard search string box and click *Search*. The Search field is not case sensitive and spaces are allowed.

   To filter by when the job(s) executed, select one of the following options:
   - Show last execution of a job.
   - Show status relative to today—Select the number of previous days over which to view job executions.
   - Show status as a set period—Type the date range or select the dates by clicking the calendar icons.
3. Click *Search* to update the list.
4. To sort the values in each column in ascending or descending order, click the column heading.
5. Find the overall status of a batch job execution by examining the indicator in the *Status* column.

Indicator	Description
✓	A green icon indicates that the batch job ran without error.
!	A yellow icon indicates that the batch job has one or more warnings.
✗	A red icon indicates that the batch job experienced an error.

   Check the End Time column to see if or when the job completed.
6. If a batch job execution has a red status, examine the trace, monitor, and error logs for more information.
7. To view detailed information about a particular job execution, look at the data on the Batch Job Status page.

   If the job includes a server group icon in the Job Server column, this indicates that the job was executed by a server group. You can roll your cursor over the server group icon to view the name of the server group. The Job Server listed is the Job Server in the server group that executed the job.

i Note

All jobs can be executed by an explicitly selected Job Server or by a server group. If you choose to execute a job using a server group, you can use this page to see which Job Server actually executed the job. If you explicitly select a Job Server to execute a job, then even if it is also part of a server group, the server group icon does not appear for the job in the Job Server column on this page.

Related Information

Management Console Guide: Setting the status interval [page 1877]

1.7.1.2 Statistics

For each job execution, the Administrator shows statistics. Statistics quantify the activities of the components of the job. You can view the following types of statistics:

- Job statistics such as time spent in a given component of a job and the number of data rows that streamed through the component.
- Data flow object statistics such as the cache size used by a transform within a data flow.

1.7.1.2.1 To view job statistics

To help tune the performance of a job, review job statistics.

1. Select *Batch* **`<repository>`**.
2. On the Batch Job Status page, find a job execution instance.

   Identify an instance using the page sub-title (which provides the name of the repository on which SAP Data Services stores the job) and the following column headings on this page:

Column	Description
Status	See Overall Status.
Job Name	The name that you gave the job in the Designer.
System Configuration	Name of a set of datastore configurations that the job uses to connect to source and target databases when it executes. Each value in this column is a link. Click the link to view the set of datastore configurations in the system configuration. To change the system configuration, click the *Batch Job Configuration* tab, then use the *Execute*, *Add Schedule* or *Export Execution Command* pages.
Job Server	The server that ran this job.
Start Time	The date and time that the job execution instance started.

Column	Description
End Time	The date and time that this job execution instance stopped.
Duration	The time (in seconds) that the job took to complete.
Run #	The number of times that this instance ran before completing.

3. Under *Job Information* for an instance, click *Monitor*.

   The Administrator opens the Job Server Monitor Log Viewer page. This page shows several statistics about this instance of job execution starting with the name of the monitor log file.

   After the file name, each line in the log provides the following information:

Column	Description
Path Name	Indicates which object (step in a data flow) is executing.
State	Indicates the run-time order of the processes in the execution of the transform object and the states of each process. These are not error status states. However, if a process state is *Proceed* and it never changes to *Stop*, this indicates the process ran with errors.
Initializing	Indicates that the job is initializing.
Optimizing	Indicates that the job is optimizing.
Proceed	Indicates that the process is executing.
Stop	Indicates that the process ended without error.
Row Count	Indicates the number of rows processed through this object. This value updates based on the *Monitor sample rate (# of seconds)* set as an execution option on the Execute Batch Job page.
Elapsed Time	Indicates the time (in seconds) since the object received its first row of data.
Absolute Time	Indicates the time (in seconds) since the execution of this entire data flow (including all of the transforms) began.

Related Information

To view overall status of executed jobs [page 102]

1.7.1.2.2 Data flow statistics

To help tune the performance of a data flow, review data flow statistics.

Related Information

Performance Optimization Guide: Measuring performance of jobs [page 2107]

1.7.1.3 To ignore error status

The *Batch Job Status* page includes an option to *Ignore Error Status*. Use this option if you are working through jobs with warnings or errors on this page and you want to mark a row so that you know you are finished looking at its logs.

1. On the *Batch Job Status* page, select the job or jobs that you want to ignore.
2. Click the *Ignore Error Status* button.

   The page refreshes and the rows you selected now display a green status icon.

1.7.1.4 Deleting batch job history data

The *Batch Job Status* page includes an option to delete information about how a job ran. If you want to manually delete rows from this page, select the rows that you want to delete, then select *Delete*. You can also manage this information by setting the Administrator's log retention period.

> i Note
>
> When you delete this job information, the software also clears data validation statistics from Data Validation Metadata Reports.

1.7.1.5 Stopping a running job

The *Batch Job Status* page includes an option to abort batch jobs. If a batch job is running and you need to stop it, select the check box next to the job name and click *Abort*.

1.7.1.6 To delete trace, monitor, and error logs for a batch job

You can view and delete trace, monitor, and error logs for job instances from the *Batch Job Status* page. The corresponding Job Server must be up and running to view or delete these logs.

You can set trace log options on the *Execute Batch Job* page.

You can use the *Delete* button on the *Batch Job Status* page to delete a set of batch log history files from a Job Server computer and its corresponding repository.

1. Select ▶ *Batch* ▶ `<repository>` ◀.
2. Select the job or jobs for which you want to delete logs.

   Alternately, you can click *Select All*.
3. Click *Delete*.
   The batch log history files are deleted from the Job Server computer and its corresponding repository.

Related Information

Management Console Guide: Batch job logs [page 1958]
Statistics [page 103]
Reference Guide: Objects, Log [page 935]

1.8 Lifecycle management

1.8.1 Migration basics

About this section

Migration as it relates to SAP Data Services is the process of moving applications through multiple development phases into production. The software supports simple and complex application migration through all phases into production.

Related Information

Development phases [page 106]
Migration mechanisms and tools [page 108]

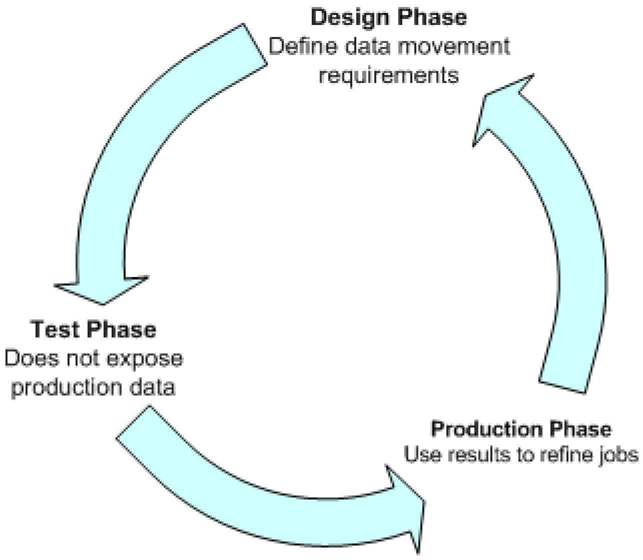
1.8.1.1 Development phases

The ETL application development process typically involves three distinct phases:

- Design phase
- Test phase
- Production phase

You can use SAP Data Services in all three phases. Because each phase might require a different repository to control environment differences, the software provides controlled mechanisms for moving objects from phase to phase.

Each phase could involve a different computer in a different environment with different security settings. For example, design and initial test may only require limited sample data and low security, while final testing may require a full emulation of the production environment including strict security.


Design Phase
Define data movement requirements
Test Phase
Does not expose production data
Production Phase
Use results to refine jobs

1.8.1.1.1 Design phase

In this phase, you define objects and build diagrams that instruct SAP Data Services in your data movement requirements. The software stores these specifications so you can reuse them or modify them as your system evolves.

Design your project with migration to testing and final production in mind. Consider these basic guidelines as you design your project:

- Construct design steps as independent, testable modules.
- Use meaningful names for each step you construct.
- Make independent modules that can be used repeatedly to handle common operations.
- Use test data that reflects all the variations in your production data.

1.8.1.1.2 Test phase

In this phase, you use SAP Data Services to test the execution of your application. At this point, you can test for errors and trace the flow of execution without exposing production data to any risk. If you discover errors during this phase, return the application to the design phase for correction, then test the corrected application.

Testing has two parts:

- The first part includes designing the data movement using your local repository.
- The second part includes fully emulating your production environment, including data volume.

The software provides feedback through trace, error, and monitor logs during both parts of this phase.

The testing repository should emulate your production environment as closely as possible, including scheduling jobs rather than manually starting them.

1.8.1.1.3 Production phase

In this phase, you set up a schedule in SAP Data Services to run your application as a job. Evaluate results from production runs and when necessary, return to the design phase to optimize performance and refine your target requirements.

After moving the software into production, monitor it in the Administrator for performance and results. During production:

- Monitor your jobs and the time it takes for them to complete.
  The trace and monitoring logs provide information about each job as well as the work flows and data flows contained within the job.
  You can customize the log details. However, the more information you request in the logs, the longer the job runs. Balance job run-time against the information necessary to analyze job performance.
- Check the accuracy of your data.

To enhance or correct your jobs:

- Make changes in your design environment.
- Repeat the object testing.
- Move changed objects back into production.

1.8.1.2 Migration mechanisms and tools

SAP Data Services provides two migration mechanisms:

- Export/import migration works best with small to medium-sized projects where a small number of developers work on somewhat independent Data Services applications through all phases of development.
- Multi-user development works best in larger projects where two or more developers or multiple teams are working on interdependent parts of Data Services applications through all phases of development.

Regardless of which migration mechanism you choose, it is recommended that you prepare for migration using one or more tools that best fit your development environment for more information). The mechanism and tools you use will depend on the needs of your development environment.

If your source data will come from multiple, homogeneous systems, it is recommended that you use Datastore and system configurations tools.

When migrating applications in a multi-user environment, it is strongly recommended that you use Naming conventions for migration.

Related Information

Export/import migration [page 109]
Designer Guide: Multi-user development [page 798]
Preparing for migration [page 110]
Datastore and system configurations [page 114]
Designer Guide: Datastores, Creating and managing multiple datastore configurations [page 239]

Naming conventions for migration [page 111]

1.8.1.2.1 Which mechanism is best?

Although SAP Data Services supports a multi-user environment, you may not need to implement this architecture on all projects. If your project is small to medium in size and only consists of one or two developers, then a Central Repository may not be a necessary solution to integrating the work of those developers.

For example, only two consultants worked on a certain HR data mart application. The Development system was designed so that while Consultant 1 managed the Master Repository, Consultant 2 worked on a new section within a complete copy of the Master Repository.

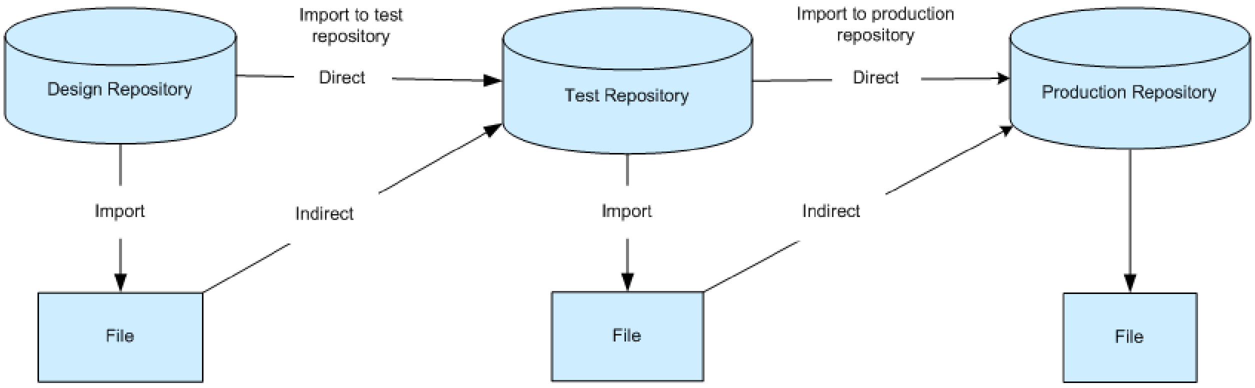
Consultant 2 then exported this new section back into the Master Repository using the export utility that allows objects to be 'Created', 'Replaced', or 'Ignored'. After updating the Master Repository, Consultant 2 took a new complete copy of the Master Repository, overwriting the previous copy.

Use the following matrix to help you determine which mechanism and tools would work best in your environment.

Situation/requirements	Migration Mechanisms		Tools	
	Export/import	Multi-user	Naming conventions	Configurations
Small to medium-sized project	X		O	O
Multiple-team project		X	X	O
Source data from multiple, homogeneous systems			X	X
Different source or target database among environments			X	X
Need a "fast and easy" migration solution	O		X	
Optimal solution: X Compatible solution: O				

1.8.1.2.2 Export/import migration

Export/import is the basic mechanism for migrating SAP Data Services applications between phases. First, you export jobs from the local repository to another local repository or to an intermediate file which you can then import into another local repository. For example, when moving from design repository to test repository, you export from the design repository to a file, then import the file to your test repository.



If you find application errors during testing, you can correct them in the development environment, then export the corrected version and import it back into the test repository for retesting.

Related Information

Export/Import [page 117]

1.8.1.2.3 Multi-user migration

You can also migrate SAP Data Services applications between phases in more complex development environments. Instead of exporting and importing applications, multi-user development provides a more secure check-in, check-out, and get mechanism, using a central repository to store the master copies of your application elements. Multi-user development includes other advanced features like labeling and filtering to provide you more flexibility and control in managing application objects.

Related Information

Designer Guide: Migrating multi-user jobs [page 824]

1.8.2 Preparing for migration

About this section

Before you develop SAP Data Services applications, it is recommended that you first set up a comprehensive structure to facilitate the migration process between development phases.

This section discusses tools that can help you build your migration structure.

It is recommended that you implement standardized naming conventions for connectivity between computer systems. Add datastore and system configurations to more easily work with multiple homogeneous systems.

Related Information

Naming conventions for migration [page 111]
Datastore and system configurations [page 114]

1.8.2.1 Naming conventions for migration

The best way to ensure fast and seamless migration is to use common naming conventions across all systems and phases of all your development environments.

Just as it is recommended that you standardize object prefixes, suffixes, and path name identifiers to simplify your projects internally, we also recommend the use of naming conventions externally for migration purposes.

To ease migration, use common naming conventions for:

- Connections to external datastores
- Directory locations
- Schema structures and owners

You want to make it as quick and easy as possible to migrate applications between users and between phases. This translates to significantly reducing or eliminating time spent reconfiguring your jobs to work in each specific environment.

While the actual data you are extracting, transforming, and loading usually differs by database, the essential structure of the data should be the same on every database with which you want the same applications to work. Therefore, it makes the most sense to standardize your database naming and structuring before starting the development process.

Related Information

Designer Guide: Naming conventions for objects in jobs [page 208]
Connections to external datastores [page 112]
Directory locations [page 113]
Schema structures and owners [page 113]

1.8.2.1.1 Connections to external datastores

Migration is the process of moving objects between local repositories, whether directly using the Export/Import method or indirectly using the Multi-user development method. Regardless of method, you must consider how the migration will impact connection configurations associated with your jobs.

Using generic naming for similar external datastore connections reduces the time you spend on reconfiguring the connections to the same database type. For example, you should choose the same logical name for all your Oracle datastore connections to the same type of database structure regardless of migration phase environment.

You can make connection names meaningful to a certain phase and specific computer system names (Test_DW, Dev_DW, Prod_DW), however if you choose this naming structure, it is recommended that you use datastore configurations for migration purposes.

Development phase	Test phase
User name: Dev_DW	User name: Test_DW
Password: Dev_DW	Password: Test_DW
Host String: Dev_DW	Host String: Test_DW

For a job to run against Test and Development, it would have to use Test_DW and Dev_DW and this would require you to create different datastore configurations for when the job runs against the Test or the Dev instance, respectively.

Alternatively, you could call the connection string DW and regardless of what instance you ran the job against, it would run without users having to create multiple datastore configurations.

Development Phase		Test Phase	
Database A	Datastore Connection	Database B	Datastore Connection
User name: DW	User name: DW	User name: DW	User name: DW
Password: DW	Password: DW	Password: DW	Password: DW
Host string: DW	Owner name: DW	Host String: DW	Owner name: DW

Examples:

- There is one Oracle source system in your company that processes order entry data. Multiple instances of this system exist for development, test, and production purposes. Therefore, you name the connection string to your Oracle source system "ORDER_SYSTEM". Then in all phases, you configure that name to point to the correct (phase-specific) instance of the system.
- Name the connection string to your target data warehouse "DW" then point it to different databases depending on whether you are in the development, test, or production environment.

When you use this generic, cross-phase naming method, you cannot access both dev and test from the same computer (since the connection string maps only to one instance). If you require access to both, use multiple datastore configurations.

Related Information

Export/Import [page 117]

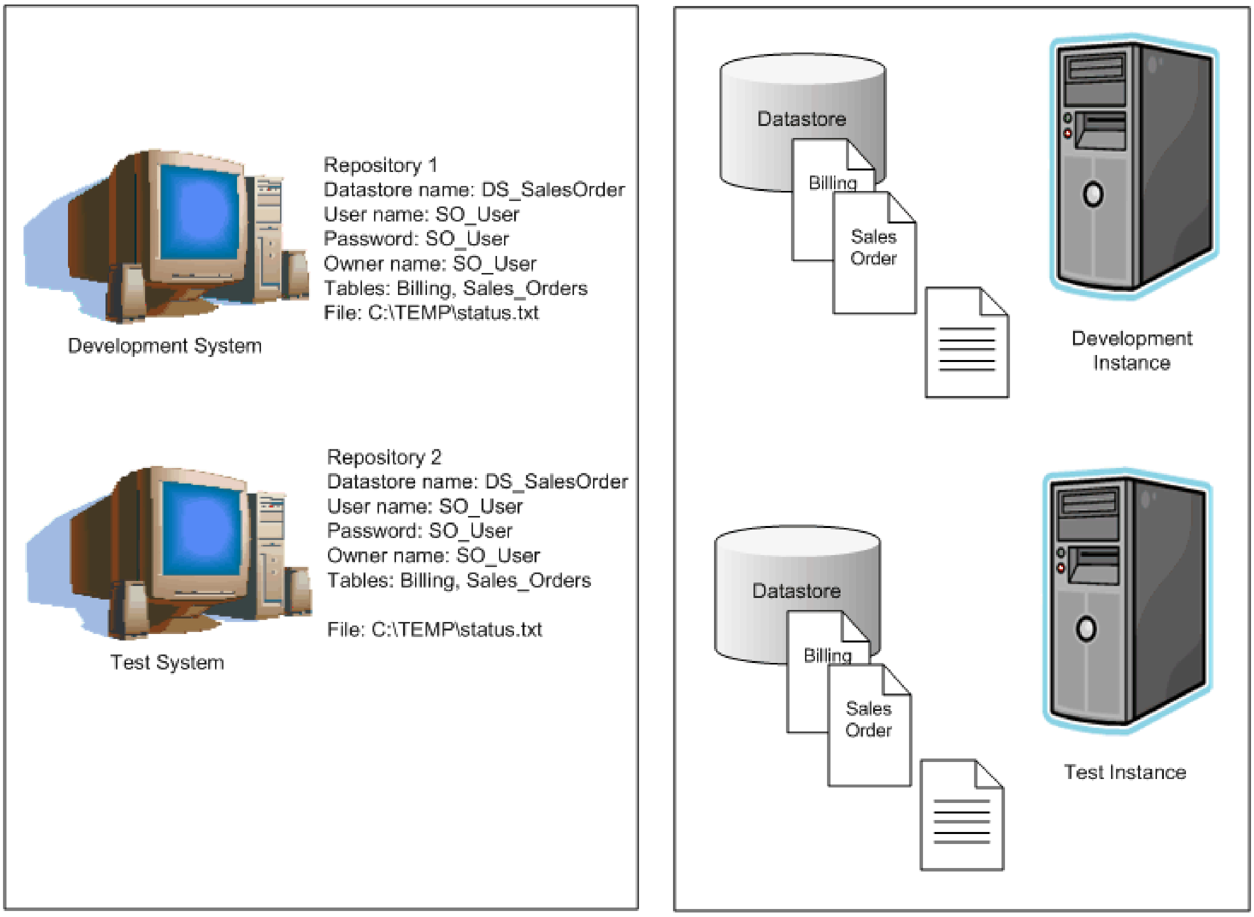
Designer Guide: Multi-user development [page 798]

1.8.2.1.2 Directory locations

It is recommended that you use logical directory names (for example, `X:\`) or point to common local drives to standardize directory location. For example, since every computer has a `C:\` drive, pointing to the directory location, `C:\TEMP` would be a safe, reproducible standard.

1.8.2.1.3 Schema structures and owners

To further facilitate a seamless structure between development phases, give all your database instances the same owner name for the same schema structures from which you are reading and to which you are loading. Regardless of name, the owner of each schema structure can vary and the software will reconcile them.



Repository 1
Datastore name: DS_SalesOrder
User name: SO_User
Password: SO_User
Owner name: SO_User
Tables: Billing, Sales_Orders
File: C:\TEMP\status.txt

Repository 2
Datastore name: DS_SalesOrder
User name: SO_User
Password: SO_User
Owner name: SO_User
Tables: Billing, Sales_Orders

File: C:\TEMP\status.txt

1.8.2.2 Datastore and system configurations

Datastore and system configurations are powerful tools for reducing the configurations required to execute the same logic against different datastore environments. With configurations, migration between development phases becomes faster and more simplified.

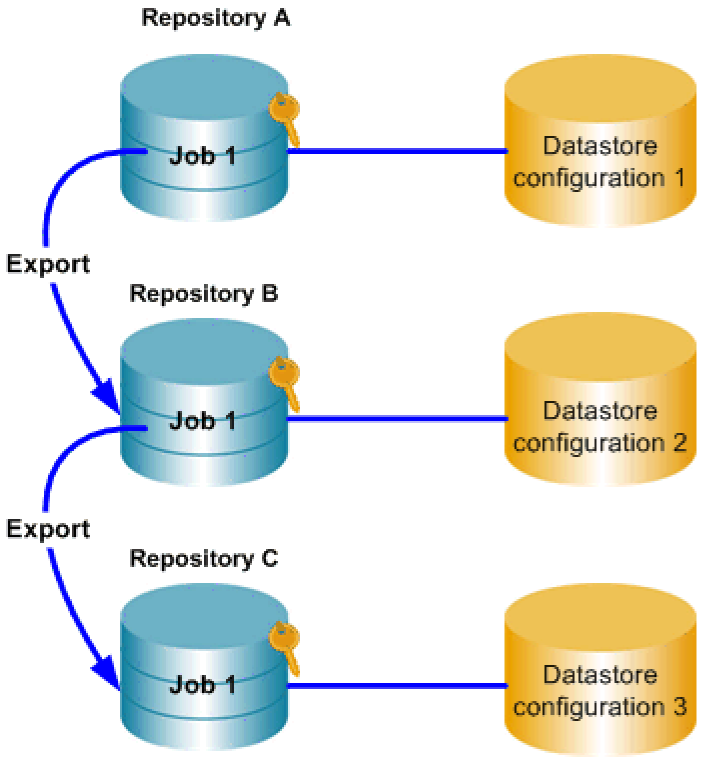
Related Information

Datastore configurations and migration [page 114]
Multiple configurations in multi-user environments [page 116]

1.8.2.2.1 Datastore configurations and migration

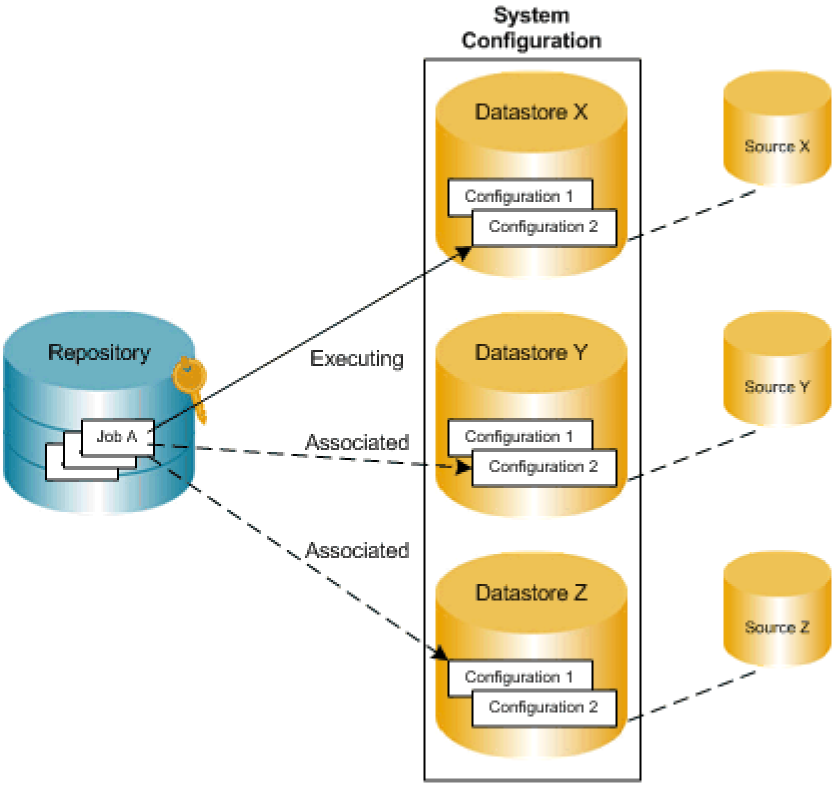
Without multiple configuration datastores, each time you export/import from one repository to another, you may need to spend time reconfiguring datastore connections to work with the new repository (and sometimes new host computer).

Without multiple configurations, each job in a repository can run only against one datastore configuration.

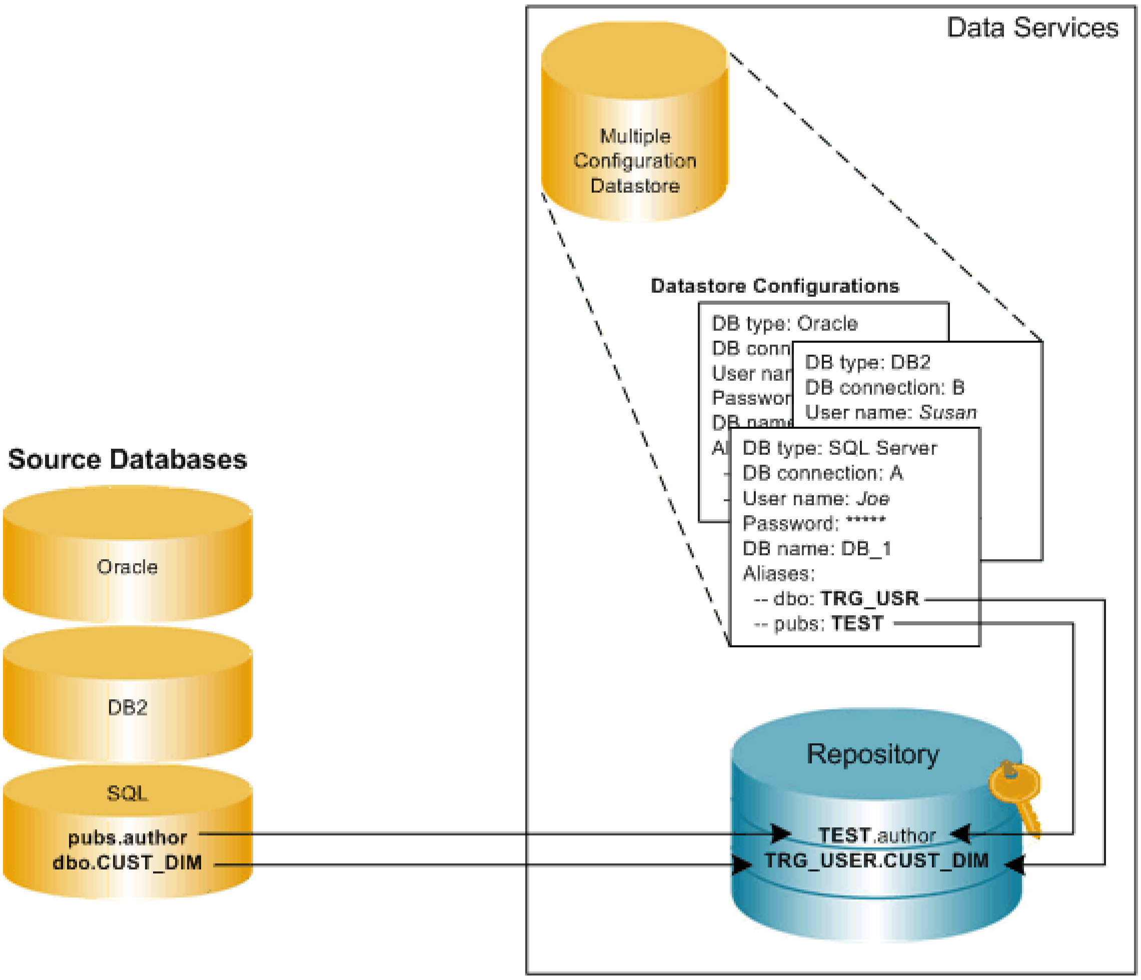

Repository A
Job 1
Datastore configuration 1
Export
Repository B
Job 1
Datastore configuration 2
Export
Repository C
Job 1
Datastore configuration 3

With multiple configurations, instead of a separate datastore (and datastore configuration) for each database instance, you can associate multiple datastore configurations with a single datastore definition.

Each system configuration defines a set of datastore configurations that you want to use together when running a job. You must create datastore configurations for the datastores in your repository before you can create system configurations.



All objects you want to import into a multiple configurations datastore must share the same owner.



Related Information

Designer Guide: Datastores, Creating and managing multiple datastore configurations [page 239]

1.8.2.2.2 Multiple configurations in multi-user environments

SAP Data Services also supports a multi-user development environment. A team can work together on an application during development, testing, and production phases. Further, different teams can work on the different phases simultaneously.

Individual users work on an application in their unique local repositories. The team uses a central repository to store, check in, and check out objects that belong to the application master copy. The central repository preserves all versions of an application's objects, allowing you to revert to a previous version if needed.

The easiest way to set up your environment to work with multi-user functionality is by establishing the exact same environment naming standards among your developers. In each developer's environment, the configuration would be different. For example a database connection string would point to their local database. However, if implementing these naming standards is not possible, you can still save time and streamline your multi-user environment by using multiple-configuration datastores.

For example, if your developers use databases with the same metadata structure but different database instances and owners, you can define a datastore configuration for each developer on your design team, mapping different owners to a common set of aliases used by all. This way, they can share and contribute to the same projects without having to set up their datastore connection information each time they check out a project from the central repository.

Related Information

Designer Guide: Multi-user development [page 798]
Designer Guide: Multi-user environment setup [page 802]
Designer Guide: Working in a multi-user environment [page 810]
Designer Guide: Migrating multi-user jobs [page 824]

1.8.3 Export/Import

Overview of export/import

The simplest type of migration in Data Services is called export/import.

This section discusses the export/import method in SAP Data Services Designer.

1.8.3.1 Exporting/importing objects

The export feature gives you the flexibility to manage and migrate projects involving multiple developers and different execution environments. When you export a job from a development repository to a production repository, you can change the properties of objects being exported to match your production environment.

In particular, you can change datastore definitions—application and database locations and login information—to reflect production sources and targets.

You can export objects to another repository or a flat file (.atl or .xml). If the destination is another repository, you must be able to connect to and have write permission for that repository, and your repository versions must match.

You cannot export read-only transform configurations.

Related Information

The Designer Export editor [page 118]
Exporting objects to another repository [page 119]
Exporting objects to a file [page 121]
Exporting a repository to a file [page 121]
Importing from a file [page 122]
Export and import options [page 163]

1.8.3.1.1 The Designer Export editor

In the Designer Export editor, you specify the objects you want to export and an export location. In Designer, choose *Tools* > *Export* or select an object and right-click *Export* to open the Export editor.

To specify an object to export, drag the object from the object library into the *Objects to Export* window.

The Object to Export window shows the final list of objects to be exported. When you drag any object from the object library, the datastores, file formats, custom functions, and transform configurations included in the object definition are automatically added to the other export sections. Each object in an export window opens to show objects called by this object.

You can control which associated objects to exclude or include. For example, you can export a work flow and all tables contained in the work flow without exporting an associated data flow.

To control which objects to export, either select an object, right-click, and choose a shortcut menu option, or select the white space in the Export editor, right-click, and choose a shortcut menu option:

Option	Description
Export	Starts the export process.
Exclude	Removes only the selected object from the list of objects to be exported. The object remains in the list, but its exclusion is indicated by a red "x" on the object icon. All occurrences of the object are excluded. When you export the list, excluded objects are not copied to the destination. Objects called by this object are not removed from the list of objects to be exported, unless they are specifically excluded. i Note You cannot export read-only transform configurations, so they are automatically excluded.
Include	Adds an excluded object to the export plan. The red "x" on the icon disappears. All occurrences of the object are included. When you export, the included objects are copied to the destination.

Option	Description
Exclude Tree	Removes the selected object and all objects called by this object from the export. The objects remain in the list, but their exclusion is indicated by a red "x" on the icons—the selected object and any objects it calls are excluded. When you export the list, the excluded objects are not copied to the destination.
Include Tree	Adds the selected excluded object and the objects it calls to the export list. The red "x" on the selected object and dependents disappears. When you export the list, the included objects are copied to the destination.
Exclude Environmental Information	Removes all connections (datastores and formats) and their dependent content (tables, files, functions) from the objects in the Export editor. Note that if you exclude datastores during export, data flows that depend on those datastores will not execute properly unless your destination repository has the same set of datastores with the same database types and versions (connection strings can be different). When you export, excluded objects are not copied to the destination. From the white space in the Export editor, right-click to select Exclude environmental information from the menu. Using this option you can export jobs without connections as a way to avoid connection errors. If you decide to use this option, it is recommended that you configure datastores and formats for the new environment separately.
Clear All	Removes all objects from all sections of the editor.
Delete	Removes the selected object and objects it calls from the Export editor. Only the selected occurrence is deleted; if any of the affected objects appear in another place in the export plan, the objects are still exported. This option is available only at the top level. You cannot delete other objects; you can only exclude them.

Related Information

Designer Guide: Datastores, Database datastores [page 210]
Designer Guide: Datastores, Creating and managing multiple datastore configurations [page 239]
Reference Guide: Datastore [page 860]

1.8.3.1.2 Exporting objects to another repository

You can export objects from the current repository to another repository. However, the other repository must be the same version as the current one. The export process allows you to change environment-specific information defined in datastores and file formats to match the new environment.

1. In the object library, choose an object to export. Right-click and choose *Export*.

The Export editor opens in the workspace. To add more objects to the list of objects to export, drag the objects from the object library into the Objects to Export section of the editor.

2. Refine the list of objects to export.

   You can use the options available in the right-click menu for each object to include or exclude the object from the export list.
3. When your list is complete, right-click and choose *Export*.
4. In the *Export to repository* window, enter your user credentials for the Central Management Server (CMS).

Option	Description
System	Specify the server name and optionally the port for the CMS.
User name	Specify the user name to use to log into CMS.
Password	Specify the password to use to log into the CMS.
Authentication	Specify the authentication type used by the CMS.

5. Click *Log on*.
   The software attempts to connect to the CMS using the specified information. When you log in successfully, the list of local repositories that are available to you is displayed.
6. Select the repository you want to use as the export target.
7. Click *Next* to continue exporting to the selected repository.
8. In *Export Confirmation* window, verify the components to export.

   The Destination status column shows the status of the component in the target database and the proposed action.

Destination Status	Action
Does not exist	Create/Exclude
Exists	Replace/Exclude

   To edit an action, select any number of objects (using the SHIFT and CTRL keys) and select either *Create*, *Exclude*, or *Replace* from the Target Status list box.
9. Click *Next*.
10. In the *Datastore Export Options* window, select the datastore, change the owner of a table or the connection properties of the datastore as necessary, and click *Advanced*.
11. Change the database connection information as required by the target database and click *Next*.
12. In the *File Format Mapping* dialog, select a file and change the Destination Root Path, if necessary.

    You can change the Destination Root Path for any file formats to match the new destination.
13. Click *Finish*.

SAP Data Services copies objects in the Export editor to the target destination. When copying is complete, the objects display in the *Output* window. The *Output* window shows the number of objects exported as well as a list of any errors.

1.8.3.1.3 Exporting objects to a file

You can also export objects to a file. If you choose a file as the export destination, Data Services does not provide options to change environment-specific information.

1. Right-click an object in the object library, and click *Export*.
   The Export editor opens in the workspace. To add more objects to the list of objects to export, drag the object from the object library into the Objects to Export section of the editor.
2. Refine the list of objects to export.

   You can use the options available in the right-click menu for each object to include or exclude the object from the export list.
3. When your list is complete, right-click the editor and click *Export to ATL file* or *Export to XML file*, depending on the type of file format that you want to export.

   > 🡆 Tip
   >
   > ATL is the software's proprietary format. Using XML might make repository content easier for you to read. XML can also be used with the object creation XML toolkit. For more information, see the *Integrator Guide*.

   By default, non-executable elements are excluded from exported XML files to improve readability. For example, the exact arrangement of transforms within a data flow would not be maintained, and the transforms would be arranged automatically when imported back into the software.

   If you want to include these elements, deselect *Exclude non-executable elements from exported XML document*. This option is available in the *Designer* > *General* group in the *Tools* > *Options* menu.
4. Specify the location for the exported file.
5. Enter the case-sensitive passphrase to use to encrypt any passwords that are stored in the objects you are exporting and click *OK*.

   > ℹ Note
   >
   > You must enter the same passphrase when you import the file back into a repository. If you use an incorrect passphrase, the software will still import the objects, but any stored passwords will be removed.

This option (*Export to XML file*) allows SAP Data Quality Management SDK developer to configure Data Quality transforms within the Data Services Designer and export the settings to XML files for use with the Data Quality Management SDK.

However, if you employ Data Services as a configuration tool for the Data Quality Management SDK, Data Services does not support the creation of a change log for changes to the configuration. You can employ the Data Services central repository concept to manage changes to the Data Quality transforms, but no change log is created.

1.8.3.1.4 Exporting a repository to a file

You can also export an entire repository to a file. When you export or import a repository, jobs and their schedules (created in SAP Data Services) are automatically exported or imported as well. Schedules cannot be exported or imported without an associated job and its repository.

If you choose a file as the export destination, the software does not provide options to change environment-specific information.

1. From the object library, right-click and choose *Repository* > *Export To File*.

   A window opens to prompt you for the destination of the export file. You can browse the directory to change the location, set the file type (XML or ATL), and enter a name for the file.
2. Click *Save*.
3. Enter the case-sensitive passphrase to use to encrypt any passwords that are stored in the repository and click *Export*.

   > Note
   >
   > You must enter the same passphrase when you import the file back into a repository. If you use an incorrect passphrase, the software will still import the objects, but any stored passwords will be removed.

   The repository is exported to the file.

1.8.3.1.5 Importing from a file

Importing objects or an entire repository from a file overwrites existing objects with the same names in the destination repository.

1. There are two ways to import repository files into another repository. Use *Tools* > *Import from file*, or in the object library, right-click and choose *Repository* > *Import from File*.

   A window opens for you to specify the file to import. You can import individual files or the whole repository using either an ATL, XML, DMT, or FMT file type. (ATL is the software's internal scripting language. DMT and FMT are files exported from the SAP Data Quality Management or IQ8 products.)
2. Select a file to import and click *Open*.
3. Enter the passphrase that was used to encrypt passwords when the file was exported and click *Import*.

   > Note
   >
   > If the passphrase does not match the passphrase you used to export the file, the import will continue, but any passwords will be emptied and need to be reset manually.
4. Perform any additional steps that may vary depending on the type of the file you are importing.
   - If you attempt to import an ATL file saved from an earlier version of SAP Data Services, a warning displays indicating that the version of the ATL file is lower than the repository version and that the ATL file you are about to import might contain objects that do not make optimal use of your upgraded repository. For example, new options for some features might not be available. To update an ATL file, import it into a repository of the same version then upgrade that repository. To abort the import, click *No*. To continue with the import, click *Yes*.
   - If you attempt to import an ATL file saved from a repository that is later than your current version, an error message displays indicating that the version of the ATL file is higher than the repository version and cannot be imported. Click *OK*.
   - If you attempt to import a DMT or FMT file, the software displays the File Format Editor to allow you to allow you to complete missing values for the properties of the file. Also, because DMT and FMT formats

support field names longer than 60 characters, you must uniquely rename any field names longer than 60 characters prior to importing the file.

1.8.3.2 Backing up repositories

Use your DBMS utilities to back up your repositories regularly. For information, refer to your DBMS documentation.

1.8.3.3 Maintaining Job Server performance

If you are designing jobs, typically you might use the same computer for your Designer, repository, and Job Server. In addition, you might use the same datastore for both your repository and your target database.

However, when you migrate your jobs into a test environment, the Job Server could move to a separate computer (typically from a Windows to a UNIX platform). The SAP Data Services Job Server computer uses source, target, and repository database client libraries to extract, transform, and load data according to a job's design. Therefore, the Job Server computer must have a database client installed for each database you are using to run a job. In addition, you can localize source and target databases using locale and code page settings.

When migrating jobs between different Job Servers verify that the code page used by each source and target database is the same as the code page set for the corresponding database client on the Job Server's computer.

The database client code page used by a Job Server on a Windows might be different from the one used on UNIX. For example, the Oracle client code page MS1252 on Windows should be changed to the ISO88591 code page on UNIX.

The software allows different code pages to be used in sources and targets. Mismatched locale settings do not cause errors and the software attempts to treat equivalent settings without any transcoding. However, mismatches may result in performance degradation from transcoding done by the software during job execution.

If your jobs do not require the use of different locales, you can increase performance by ensuring that default locales are not mismatched. After migration, if you notice a significant difference between the speed of design and test environments, check locale settings. In the Designer, check to see that datastore code pages for sources and targets match client code pages on the Job Server computer.

Related Information

Reference Guide: Locales and Multi-Byte Functionality [page 1793]

1.8.4 The Enhanced Change and Transport System

The Change and Transport System (CTS) transports changes between SAP systems in your system landscape. The Enhanced CTS (CTS+) promotes non-SAP (non-ABAP) contents across repositories, i.e. enables you to transport application objects between systems in your system landscape, if required, alongside ABAP objects. You can administer non-ABAP systems in a CTS transport domain in SAP NetWeaver Application Server ABAP. You transport these objects in a transport requests. When you run imports in Transport Management System (TMS), the system performs the appropriate copying of objects in an automatic and controlled manner.

The enhanced CTS functions are available with Support Package Stack (SPS) 15 of SAP NetWeaver 7.0. You also require an SAP Application Server Java with the same Support Package level.

For more information, see *SAP Note 1003674*

1.8.4.1 Transporting changes: Business context

Very often Data Services is installed in multi-tier system landscapes. Typically the Data Services customer development is done in a development Data Services system, the changes then are consolidated in a test/consolidation Data Services system and at the end the changes are brought to the production Data Services system.

To support controlled transport from the development system to the follow-up systems the SAP NetWeaver CTS transport management system was developed. With Enhanced CTS (CTS+) this is extended to Non-ABAP transports, for example for Data Services change files.

The integration of Data Services CTS+ transport management allows to provide all development changes made in the Development System to the Quality System and then to the Production System in a system controlled way.

With the transport of Data Services changes using a CTS+ system the following goals are achieved:

- Trace changes performed in an application system landscape.
- Avoid multiple erroneous manual modifications on different dependent repositories, when changes are done in a development system, tested in a test system and used in a production system.
- Transport changes through a customer-defined multi-tier application system landscape is possible.
- Guarantee similarity or even equality of all systems on the transport route in the application system landscape, regarding the application customer development.

1.8.4.2 Background information

Change and Transport System: Overview (BC-CTS)

The first reference to be consulted is the standard Change and Transport system documentation - CTS Reference Manual: *http://help.sap.com/saphelp_nw70/helpdata/EN/3b/dfba3692dc635ce10000009b38f839/frameset.htm*

The SP stack levels of the CTS system mentioned in this guide refer to the SP stacks of SAP NetWeaver. Keep in mind that SP stack levels for SAP Solution Manager are different and do not contain the same functionality as an SP stack for SAP NetWeaver. Take a look at the basis release and SP stack of SAP NetWeaver that your Solution Manager is using.

SAP Note: 1003674 Central Note on enhanced CTS.

Transporting Non-ABAP Objects in Change and Transport System

The documentation on non-ABAP Transports in the Change and Transport System can be found in the following manual: *http://help.sap.com/saphelp_nw70/helpdata/en/45/EC25370FDC3481E10000000A1553F6/frameset.htm*

Configuring TMS

Information about configuration of the transport management system (TMS) you find here: *http://help.sap.com/saphelp_nw70/helpdata/en/44/b4a09a7acc11d1899e0000e829fbbd/frameset.htm*

Transport Organizer Web UI

The Transport Organizer Web UI is used to manage the CTS+ change requests. Read the CTS+ User Reference Manual – Transport Organizer Web UI: *http://help.sap.com/saphelp_nw70/helpdata/EN/46/028ec7469204abe10000000a114a6b/frameset.htm*

CTS+ Command Line Tool

The CTS+ command line tool allows handling CTS+ requests from a command batch file. For details about the use, see SAP Note 1278181.

How-To Guide: Best Practices for Implementing CTS+

This document provides an overview about the CTS+ configuration and the CTS+ landscape setup. *https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/10456aac-44f7-2a10-1fbe-8b7bcd7bcd58*

Solution Manager 7.0 Change Request Management (ChaRM)

Change Request Management (ChaRM) enables you to manage your maintenance, implementation, template, and upgrade projects: Starting with change management and project planning, through resource management and cost control, to physical transports of changes from the development environment into the productive environment. The processes supported by Change Request Management include urgent corrections for implementing fast and direct changes in the productive environment, and activities for maintenance projects, and implementation, upgrade, or template projects. Cross-system and cross-component changes are supported.

ChaRM is the logical management of all changes, for example the transport requests of PCM systems. This includes the management and control of periods or situations, in which transport requests are approved and imported into target systems. Change Request Management works with the underlying TMS/CTS system. It also defines and controls emergency correction procedures and personnel responsible and authorized to participate in the change processes etc.

More information: *http://help.sap.com/saphelp_sm40/helpdata/en/0c/5b2160f6fa4b83a3674a210b1cdeb0/frameset.htm*

More information on the SAP Solution Manager: *http://help.sap.com/saphelp_sm40/helpdata/en/45/51fbdbd4941803e10000000a1553f7/frameset.htm*

1.8.4.3 Setting up your Data Services change files

You can send files, exported from Data Services, through the CTS+ system. These files can be in XML or ATL format, and can represent any of the following objects:

- Repositories
- Jobs
- work flows
- Datastores
- Transform configurations
- Any other object in the Object Library

These files will need to be exported to a directory that CTS+ will be able to access during import to the CTS+ system. Similarly, you may also want to create a directory that will house the files you will receive from CTS+ that you import into your repository.

You will use the normal Data Services import and export procedures to produce these files and update your objects. You may also want to implement a file naming convention to help keep track of the files.

Example

Creating a source directory

You may want to create a directory structure in the Data Services install location, such as **<LINK_DIR>**`\cts\source` to specify that the folder contains files ready to be output to CTS+. When you set up a source system in CTS+, you will point to this directory.

Example

Creating a target directory

You may want to create a directory structure in the Data Services install location, such as **<LINK_DIR>**`\cts\target` to specify that the folder contains files received from CTS+. When you set up a target system in CTS+, you will point to this directory.

Related Information

Export/Import [page 117]

1.8.4.4 Configuring the Transport Organizer Web UI

CTS+ provides an ABAP Web Dynpro application called the Transport Organizer (CTS_BROWSER) that you use to create transport requests and to attach transportable objects. You must perform configuration steps to run and use this application.

For more information, see: *http://help.sap.com/saphelp_nw70/helpdata/en/ea/6213584a3f49119eccd7d739e55d5d/frameset.htm*

Activate all of the services that are required to run ABAP Web Dynpro applications as well as the web service CTS_BROWSER using transaction SICF (Maintain Services). This includes all services for ABAP WDA outlined in *SAP Note 517484* and *SAP Note 1088717*. If after the activation of services the Transport Organizer cannot be called and the response displays error messages as outlined in *SAP Note 1088717* (for example, Service is not active or equivalent), activate the services displayed in the error messages and retry.

1.8.4.4.1 Configuring the Transport Landscape

Create the systems of your Data Services system landscape as non-ABAP systems in TMS. Follow the steps outlined in this documentation. In TMS, as is true for any SAP ABAP and SAP Java based system, systems are represented by a three alphanumeric character identifier, called system identifier or SID. This SID is representing the system not only in TMS, but also in other managing applications, for example the SAP System Landscape Directory, the SAP Solution Manager, etc. Please provide your Data Services with a SID, for example DSD for the development system, DSQ for the test system, DSP for the production system. If you use the System Landscape Directory (SLD) to register Data Services systems, please use the same SIDs as reported to SLD. Systems are thereafter connected to so called transport routes. They provide a clear understanding to the Data Services administrators who are using the Transport Organizer Web UI and TMS to manage and control the transport requests.

For details, refer to reference manual (Defining and Configuring Non-ABAP Systems): *http://help.sap.com/saphelp_nw70/helpdata/en/45/f64a3dbc1a04a9e10000000a114a6b/frameset.htm*

1.8.4.4.1.1 Define the System Landscape

1. Log on to the CTS+ system and start transaction STMS (Transport Management System).
2. Choose *System Overview* to get the list of all systems defined in the CTS system.

1.8.4.4.1.2 Create the source system

In this step, you will need to define Data Services as a new non-ABAP System.

1. Choose ▶ *SAP System* ▶ *Create* ▶ *Non-ABAP System.* ◀
   The *TMS: Configure Non-ABAP System* window is displayed.

2. Create the Data Services System with a system ID (using the system's SID) and a description
3. Select the CTS+ system as the communication system.
4. Define the Data Services system as a source system by selecting the *Activate Transport Organizer* option.
5. Enter the client where you want to use the Transport Organizer.
6. Save your settings and confirm that you want to distribute the TMS configuration.
7. Add location information for the common file share for Data Services output/CTS+ inbox by selecting the newly created system from the list and double-clicking it.
8. In the *Transport Tool* tab, add the additional parameter: NON_ABAP_WBO_INBOX for the development system.
   For example, **`<LINK_DIR>`**`\cts\source`.

   > **Note**
   >
   > Alternatively, you may upload rom a local machine (client) in the Transport WebUI.

   For more information, see *http://help.sap.com/saphelp_nw70/helpdata/EN/6f/90813e26b1443d9d3642bb5cd8234c/frameset.htm*

1.8.4.4.1.3 Create the target system

In this step you will create a target systems (test and production) in the same way you created the source system.

1. Choose *SAP System* > *Create* > *Non-ABAP System.*
   The *TMS: Configure Non-ABAP System* window is displayed.
2. Create the Data Services System with a system ID (using the system's SID) and a description.
3. Select the CTS+ system as the communication system.
4. Define the Data Services system as a target system by selecting the *Activate Deployment Service* option.
5. Select the *File* checkbox as your preferred deployment method.
6. In the *Directory* field, enter the file share where CTS+ is providing the change data to the target system.
   For example, **`<LINK_DIR>`**`\cts\target`.
   This value then appears in the DEPLOY_OUTBOX parameter in the *Transport Tool* tab. For more information, see *http://help.sap.com/saphelp_nw70/helpdata/EN/2e/674953194c4299abae253152544fab/frameset.htm*.
7. Save your settings and confirm that you want to distribute the TMS configuration.

1.8.4.4.2 Defining transport routes

You systems are now ready to be included in a transport route in CTS.

To define transport routes, you will use the Graphical Editor in STMS. To do this, log on to Domain Controller (in client 000), start transaction STMS, and click the *Transport Routes* icon.

Create one standard transport layer, which is the default.

Using the Graphical Editor for definition of transport routes is explained in the following manual: *http://help.sap.com/saphelp_nw70/helpdata/en/44/b4a2a27acc11d1899e0000e829fbbd/frameset.htm*.

Now the Data Services systems are defined in the transport system landscape, and a transport route is defined from a development Data Services system to a production Data Services system.

1.8.4.5 Providing changes to CTS+ transport system

After the changes performed in the Data Services development system are extracted (exported), the next step is to provide the change file to the CTS+ Transport Control System. To do this, you will need to create a new transport request and import the change file into the new CTS+ transport request. You can do this in one of two ways:

- Using the Transport Organizer Web UI
- Using the CTS+ Command Line Tool

> Note
>
> The installation and setup of the CTS+ system and the CTS+ Transport Organizer Web UI are not part of this manual. Links to this documentation (including the CTS+ command line tool) can be found in the *Background information* [page 124] topic.

1.8.4.5.1 Change file attachment with CTS Transport Organizer Web UI

The CTS+ Transport Organizer Web UI is the UI that allows you to create and administrate Data Services change requests. It also provides the functionality to attach the change file to the change request.

For sharing with CTS, you need to define a shared folder on the CTS host where exported Data Services change files can be accessed from the CTS host. In this documentation, we use the following example share for data exchange: `\\`**`<CTSServer>`**`\DSOutbox` - a shared folder that is physically stored on the same server where the CTS application server is implemented. This folder is used as shared folder for the source Data Services change extraction output. (In customer installations this should be the **`<LINK_DIR>`**`\cts\source` folder)

1.8.4.5.1.1 Provide change data via NON_ABAP_WBO_INBOX

The change files from **`<LINK_DIR>`**`\cts\source` have to be accessible by CTS+ by sharing this folder either with the CTS+ system or with your local desktop. To directly access it from CTS host the parameter NON_ABAP_WBO_INBOX has to be defined to point to the share where Data Services puts its exported data (Parameter NON_ABAP_WBO_INBOX == \\<CTSServer>\DSOutbox).

1.8.4.5.1.2 Start the Transport Organizer Web UI

1. Log in to the Domain Controller, within the client as specified when creating the source system.
2. Start transaction STMS.
   Now the Transport Organizer Web UI can be started via Environment > Transport Organizer Web UI.
3. Enter the SID of the Data Services source system.

1.8.4.5.1.3 Create a new transport in CTS

1. Click the *Create Request* button to begin the creation of the new CTS+ change request.
2. Enter a short description and check for the correct owner.
3. Click *Create*.

A new transport request is created, and it appears in the *Requests* column.

1.8.4.5.1.4 Attach change file to the transport request

1. Select the transport request, then click the *Object List* tab to see if there are already attached objects.
2. To attach the change file, click the *Attach* button.
   The *Attach file object* window appears.
3. Select *Other* in the *Application* option.
4. Select *Client* if the change file is on the same local client that the Transport Organizer Web UI is running. If the Data Services Outbound folder is shared on the CTS server, select *Server*.
5. Click *Browse* to find the shared file and to upload it to the CTS+ system.
6. Select the correct change file, and click *Open*.
7. Click *OK* to upload the file to the transport request and to CTS.

1.8.4.6 Transport in the System Landscape

After you have attached all of the Data Services files that you want to transport with one transport request, you have to release your transport request and start the import to the target system, which is the next system in your transport route. During the import, the files are copied to the Data Services inbound folder.

In the CTS+ configuration, you can specify whether the Import Queue is handled automatically or if the queued transports have to be imported manually.

The deployment of a Data Services-specific transport orders in the current version is done with provisioning the change files to the CTS+ outbox folder defined for the Data Services target system, where the Data Services administrator has to pick it up for manual deployment.

1.8.4.6.1 Release transport request

1. Open the Transport Organizer Web UI.
2. Select the transport request to be released.
3. Click the *Release* button.

The transport request is now assigned a status of *Released*.

> **Note**
>
> If any issues occur with the release process, you may find helpful details in the Logs tab in the Transport Organizer Web UI.

1.8.4.6.2 Starting the import from the Import Queue

Processing the import will send the attached change file to the CTS outbox (or Data Services Inbound from a Data Services perspective, parameter DEPLOY_OUTBOX).

1. Start the transaction STMS, and click the *Import Overview* icon.
2. Select the target system to see the import queue for this target sytem. In the selected row, you see the icon (with a plus sign) that specifies that there are imports to be processed.
3. Double-click the target system to open the import queue.

   If you do not see the new transport request, click *Refresh*.
4. Select the transport order, and click the *Import Request* icon.

   Check the CTS+ DEPLOY_OUTBOX folder in the target sytem before and after the import and you will see a new directory with the name of the transport order.

   The*Import Transport Request* window appears.
5. Select an option to import immediately or at a later time.
6. Click *Yes* to import.

Because a file sharing between the CTS+ outbound and the Data Services Inbound has been defined for the target system (CTS+ Parameter DEPLOY_OUTBOX), the imported change file in now also accessible on the file share server you assigned.

If you encounter issues when importing, you will find details in import logs. You can mark the request in question and choose *Request* > *Display* > *Logs* .

For more information about performing transports, see *http://help.sap.com/saphelp_nw70/helpdata/en/44/b4a3507acc11d1899e0000e829fbbd/frameset.htm*.

1.8.5 Data Services Object Promotion Management

1.8.5.1 About object promotion

1.8.5.1.1 Overview of Data Services Object Promotion Management

The *Object Promotion* tool in the Data Services Administrator enables you to move one or more Data Services objects from a development environment to a QA environment, or directly to a production environment. To ensure security, these environments typically do not have direct access to one another. Object promotion in Data Services is accomplished over secure FTP or over a shared directory that only an Administrator or a user with an account assigned to the Data Services Administrator group can configure for this purpose, maintaining security in both source and target environments.

Users with *View* access to a repository can export its objects from a development environment to a shared directory created by authorized users. Users who have *Full Control* access to the repository can import objects after an Administrator or a user with an account assigned to the Data Services Administrator group has granted them the new *Manage Object Promotion Import* user application right.

> Note
>
> The user who starts the Server Intelligence Agent (SIA) on the server that has access to the shared directories that are used for exporting and importing promoted objects must have full read and write access to those shared directories. If the server is started by a user who does not have full access to the shared directory, all configurations for export and import operations for that shared directory will fail validation, and the configurations cannot be completed until the issue is resolved. To validate this, an authorized user or the Administrator can validate the configuration by testing the connection or re-saving the configuration to check. If another user without full read write privileges to the shared directory starts SIA after a configuration has been successfully validated, all export and import operations from that configuration will fail until SIA is restarted by a user with the required credentials.

If objects with the same name and types are selected for import, the objects are imported sequentially based on the date of export. When you view the list of exported objects in the *Import Configuration* page in the Administrator, the exported objects are grouped together with objects of the same type and object name, listed in order by the date they were exported. If the same object and types are exported multiple times, depending on the export date, you can import these objects the same objects and these objects will be imported sequentially, based on the export date.

1.8.5.1.2 Requirements for object promotion

The following requirements apply when using the SAP Data Services Object Promotion Management tool in the *Administrator* in Management Console to promote Data Services objects:

- The source and target repositories must both be running the same version of Data Services before attempting to promote objects to or from them.

- The user who starts the Server Intelligence Agent (SIA) on the server that has access to the shared directories that are used for exporting and importing promoted objects must have full read and write access to those shared directories.
  If the server is started by a user who does not have full access to the shared directory, all configurations for export and import operations for that shared directory will fail validation, and the configurations cannot be completed until the issue is resolved. To validate this, go to configuration page and click *Test*, or try to save the configuration again.
  If another user without full read write privileges to the shared directory starts the SIA on the server after export and import configurations have been successfully validated and saved, all of those export and import configurations will fail to successfully run until the SIA is restarted by a user with the required credentials.
- When a Data Services object is promoted, all of its dependent objects are also promoted, with the exception of its datastores. Typically, datastores in a development environment contain data that is used only for testing, and that data is not typically used in a production environment. If you need to promote a datastore from a development environment to a production environment, the datastore must be promoted separately.
- When a datastore is promoted, its password is not included in the import or export operation, so the password field for the promoted datastore is left blank. This is intentional and provides a safeguard to prevent the unintended use of a development or QA datastore in a production environment. It also provides another level of security, because only the owner of a datastore should know its password.
  After a datastore promotion is completed, be sure to reconfigure the datastore under *Administration > Management > Datastore* in SAP Data Services Management Console. Set a password for the promoted datastore in its new location, and if needed, modify the location of the database it uses. If the datastore used in the production environment is different from the one used in development or QA, be sure to update them. If a valid datastore password is not entered after the datastore has been imported into another environment, all of the jobs that depend on that datastore will fail to run until you enter the password.
- Only a user whose account is assigned to the Data Services Administrator group account can create and edit the configurations used for exporting or importing objects. An Administrator or a user who is assigned to the Data Services Administrator group can assign to another user who is not part of that group the *Manage Object Promotion Configurations* right to allow that user to modify the configurations through the Administrator in Master Console.
- Only users with, at a minimum, *View* access to a repository can export its objects to the shared directory configured by an Administrator or a Data Services Administrator group user.
- Only Data Services Administrator group users can import objects into a repository. An Administrator or a user whose account is a member of the Data Services Administrator group can assign another user who is not part of that group the right to *Manage Object Promotion Import*, which grants that user the authorization to import objects into a repository. That user must also have *Full Control* access to the repository to be able to import its objects.
- Each top-level object import is run sequentially. If one object fails to import, it will revert back, then the import continues to the next top-level object.

The following table summarizes the required user rights for each stage of the Object Promotion Management process.

Type of user	Configure export and import	Export objects from repository (source)	Import objects from shared directory (target)
Administrator	Yes	Yes	Yes
Data Services Administrator group	Yes	Yes	Yes
Regular Data Services user, without additional rights	No	Yes	No

Type of user	Configure export and import	Export objects from repository (source)	Import objects from shared directory (target)
		Must also have, at minimum, *View* access to export objects from the repository	
Regular Data Services user, with *Manage Object Promotion Configurations* right	Yes	Yes Must also have, at minimum, *View* access to export objects from the repository	No
Regular Data Services user, with *Manage Object Promotion Import* right	No	Yes Must also have the required *Full Control* access to the shared directory	Yes

Related Information

Managing application rights for a group [page 51]

1.8.5.1.3 Object promotion user interface

The Data Services Object Promotion Management tool is found in the Management Console, under the *Administrator*. To view the choices, select or expand the *Object Promotion* menu.

When you select the title for the *Object Promotion* menu, the *Object Promotion* page appears, with choices to access and run existing export and import configurations.

- *Export objects* manages the export of specific object types from specific repositories. After you select a repository and object type, and select *Next*, a table appears with details for each object of the selected type that resides on that repository and is available to export.
- *Export Substitution Parameters* manages the export of one or more substitution parameters, which are listed by the repository in which they reside. After you select a substitution parameter and select *Next*, a table appears with details for each substitution parameter.
- *Export System Configurations* manages the export of one or more individual system configurations from the local repository. After you select a system configuration and select *Next*, a table appears with details for each system configuration.
- *Import* displays details for objects that have been exported from the repository and are available to import. A search tool provides a way to narrow the displayed list. From the displayed results, you can select one or more objects to import.

Each of the pages displayed for these choices include an indication of whether an export or import has ever been run on the displayed object(s).

Field	Value = Yes	Value = No
Exported	One or more export operations have been run on this repository object. The status of these operations could be success or failure.	No export operations have ever been run on this repository object.
Imported	One or more import operations have been run on this repository object. The status of these operations could be success or failure.	No import operations have ever been run on this repository object.

When you expand the *Object Promotion* menu, there are two choices:

- *Export Configuration* lists all of the export configurations that have been created by authorized users.
- *Import Configuration* lists all of the import configurations that have been created by authorized users.

The export and import configurations are displayed and are available for selection in the list on the *Object Promotion* page, which is displayed when you select either choice.

> i Note
>
> Users whose accounts are not members of the Data Services Administrator group or who have not been granted specific permissions by a member of that group cannot create or edit export and import configurations, nor can they run the import configurations created by an authorized user.
>
> - To edit an export or import configuration, users outside the Data Services Administrator group must be granted the *Manage Object Promotion Configurations* right.
> - To run an import configuration to import objects, users outside the Data Services Administrator group must be granted the *Manage Object Promotion Import* right, and they must also have *Full Control* access to the repository.

1.8.5.2 Configuring object promotion

1.8.5.2.1 Setting up an export configuration

The following specific requirements and recommendations apply when setting up an export configuration:

- Only a user whose account is assigned to the Data Services Administrator group account can create and edit the configurations used for exporting or importing objects. An Administrator or a user who is assigned to the Data Services Administrator group can assign to another user who is not part of that group the *Manage Object Promotion Configurations* right to allow that user to modify the configurations through the Administrator in Master Console.
- For security reasons, only a system administrator can create a shared directory to which objects can be exported to or imported from. If a shared directory has not yet been created but you have been granted permission to configure exports and imports, the only available transport option you will see for your configuration is FTP.
- If you are configuring an export to use FTP to promote objects to a UNIX system, be sure to configure the FTP server to have full read and write access to the shared directory that was set up to receive exported objects.

- If you need to clear space on the shared directory, you can safely delete the exported .ATL and .manifest files after you have finished importing them. If you create a separate export configuration for each release, it will be easier to identify the groups of files you can safely archive or delete.
- A repository can be associated with multiple, different export configurations.
- For information about the general requirements for object promotion, see the related topics.

You can use object promotion to create a configuration than can be used to export an object and its dependent objects over secure FTP or directly to a shared directory:

1. In Data Services Management Console, expand the *Object Promotion* menu, then select *Export Configuration*.
2. In the *Export Configuration* page, click *Add*.
3. To create an export configuration to transport objects over FTP:
   a) On the FTP tab, enter information in each of the required fields.
   b) Choose whether to enable Secure FTP.
   c) Select one or more entries in the *Available Repositories* list to associate with this configuration. You can only export objects from the repositories that are selected for the *Associated Repositories* list.
   d) Optionally, click *Test* to ensure that the configuration can access the repositories and the *Target Directory* you associated with this configuration.
   e) Save the configuration.
4. To create an export configuration to transport objects directly to an existing shared directory:
   a) On the *Shared Directory* tab, enter a name for the configuration and a target directory to which the objects should be exported on the shared directory.
   b) Select one or more entries in the *Available Repositories* list to associate with this configuration. You can only export objects from the repositories that are selected for the *Associated Repositories* list.
   c) Save the configuration.

When you test or save the configuration, the Object Promotion Management tool validates whether the SIA service can save (read and write) content to the shared directory. If the configuration can access the repositories with which it has been associated and the SIA service validates access to the shared directory, the configuration is saved with the name you specified. After it is saved, the named configuration will appear as a choice on the *Export Configuration* page, and as an *Object Name* choice when you choose *Export Objects*, *Export System Configuration*, or *Export Substitution Parameters*.

To modify an export configuration, return to the *Export Configuration* and select the *Object Name* of the configuration you want to modify.

To remove an export configuration, select it in the *Export Configuration* and click *Remove*.

Related Information

Requirements for object promotion [page 132]

Exporting an object [page 137]

1.8.5.2.2 Setting up an import configuration

The following specific requirements apply when setting up an import configuration:

- The object and all of its dependent objects must have already been exported before you run the import configuration.
- An import configuration can be associated with only one repository.
- A target repository must be configured to a shared directory where ATL files have been exported.
- Only a user whose account is assigned to the Data Services Administrator group account can create and edit the configurations used for exporting or importing objects. An Administrator or a user who is assigned to the Data Services Administrator group can assign to another user who is not part of that group the *Manage Object Promotion Configurations* right to allow that user to modify the configurations through the Administrator in Master Console.
- For information about the general requirements for object promotion, see the related topics.

You use object promotion to create a configuration than can be used to import an object and its dependent objects from a shared directory:

1. In Data Services Management Console, expand the *Object Promotion* menu, then select *Import Configuration*.
2. On the *Import Configuration* page, click *Add*.
3. Select the *Repository* from which the objects were configured to be exported, and in the *Source Directory* field, enter the directory path to where the export configuration deposited the objects.

   After you configure the *Repository*, it cannot be configured with a different path, because that repository will no longer show up in the list of choices.
4. Save the configuration.

If the import configuration can access the shared directory to where the objects are exported, the configuration is saved with the name of the target repository. The configuration will appear as a choice on the *Import Configuration* page, and as an *Object Name* choice when you choose *Import*.

To modify an import configuration, return to the *Import Configuration* page and select the *Object Name* of the configuration you want to modify.

To remove an import configuration, select it in the *Import Configuration* and click *Remove*.

If you imported a datastore, after it is imported you must reconfigure the datastore under *Administration > Management > Datastore* in SAP Data Services Management Console and assign it a password.

After you import a collection of substitution parameters, if needed, you can update the paths for the substitution parameters after they are imported to their new location.

Related Information

Requirements for object promotion [page 132]
Importing an object [page 140]

1.8.5.3 Promoting objects

1.8.5.3.1 Exporting an object

The following specific requirements apply when exporting an object:

- The source and target repositories must be running the same version of SAP Data Services.
- You can export from a local repository or a central repository.
- If a datastore is a dependent object of the object you are exporting, the datastore must be exported separately.
- When a datastore is promoted, its password is not included in the import or export operation, so the password field for the promoted datastore is left blank. This is intentional and provides a safeguard to prevent the unintended use of a development or QA datastore in a production environment.
  After a datastore promotion is completed, be sure to reconfigure the datastore under *Administration > Management > Datastore* in SAP Data Services Management Console. Set a password for the promoted datastore in its new location, and if needed, modify the location of the database it uses.
- The repository you are exporting objects from must be associated with an export configuration.
- For information about the general requirements for object promotion, see the related topics.

You can use object promotion to export the following types of objects from their current repository to a shared directory:

- Project
- Job
- Work flow
- Data flow
- Function
- File format
- Datastore
- System configurations
- Substitution parameters

1. In Data Services Management Console, select *Object Promotion*.
2. Select one of the following choices in the *Objects* list, then click *Next*.
   - Export objects
   - Export system configurations
   - Export substitution parameters
3. Select a repository (local or central) and if you chose *Export objects*, choose an object type, then click *Next*.
   The *Export* page appears, with a list of the latest version of all the objects you specified that are available in your chosen repository.

   There are 23 entries listed on each page. You can include up to 23 of the entries listed on a given *Export* page in a single export operation. Click the page links to view any subsequent pages of available objects.
4. If you chose *Export system configurations* or *Export substitution parameters* in the *Objects* list, to narrow the list to display only the objects you want to export, use the *Search* field to look for objects that completely or partially match the export configuration for the objects you want to export. The *Search* field supports wildcards.
5. If you chose *Export objects* in the *Objects* list, to narrow the list to display only the objects you want to export, filter the list:
   a) Enter specific text to search for that appears in the *Object name* field. You can use an asterisk as a wildcard to expand your search to objects that partially match or are part of a common naming scheme.
   b) If the selected *Object Type* is *Job*, select a specific *Project* (the default is *All Projects*).
   c) If you are exporting from the central repository, additional filter choices are available:

If the selected *Object Type* is *Project*, select a different version in the *Get* list (the default is *Latest Version*).

When you submit the query, it matches the parent object's label with the associated version. If the label for a child of an object does not match with the specified label, when a parent object is exported, the latest version of the child object will be exported.

To ensure that you have total control over which version of an object's dependencies are exported, be sure to always label your objects when you check them in. Label child objects with the same label you assigned to the parent object.

d) Click *Search* to use your selections to search for objects that match your choices.

6. Individually choose up to 23 *Object Name* entries displayed on the current page, or choose *Select All* to import all 23 objects on the currently displayed page.
7. Choose the export method.
8. Click *Export* to start exporting the objects you selected.

For each object, an ATL file and a manifest file are written to the *Target Directory* specified in the export configuration.

A *Confirmation* page displays progress for the export, and the *Status* is automatically refreshed every five seconds. When the export is completed, the automatic refresh ends, the ending *Status* information is displayed (success or failure), and log information is stored in the repository under the *AL_EXPORT_HISTORY* table. To view the log information while you have the *Confirmation* page open, select *view log* under the *Status* column of the displayed report. Details about the job are on the *Trace* tab. If there were any issues with the export, messages are recorded to the *Error* tab.

A unique confirmation ID is assigned to each export attempt. The confirmation ID is used for a unique ID in the *AL_EXPORT_HISTORY* table. The confirmation ID is used to pass information when you import the exported objects. The confirmation ID contains the following information:

```
<Timestamp-in-milliseconds>-<six-digit random number>-<number of export attempts>
```

To convert the time stamp that is displayed in milliseconds to date format, copy the integers displayed next to *Confirmation#* up to the first dash, and paste them into any available conversion tool.

> **Note**
>
> Make a note of the unique confirmation number displayed on the *Confirmation* page, and copy any information from the log that you want to preserve. You cannot return to this *Confirmation* page after you dismiss it. A copy of the log is also written to the repository under the *AL_EXPORT_HISTORY* table, so you can query the table to find the log information later.

If you exported a datastore, after it is imported you must reconfigure the datastore under *Administration > Management > Datastore* in SAP Data Services Management Console and assign it a password.

If you exported substitution parameters, if needed, you can update the paths for the substitution parameters after they are imported to their new location. To edit substitution parameters, you must either be an Administrator or your account is a member of the Data Services Administrator group, or either of these users can grant you the *Manage datastore and substitution param configurations*.

Related Information

Requirements for object promotion [page 132]
Setting up an export configuration [page 135]
Managing application rights for a group [page 51]

1.8.5.3.2 Importing an object

The following requirements apply when importing an object to another repository:

- The source and target repositories must be running the same version of SAP Data Services.
- Your user account must be a member of the Data Services *Administrator* group, or a member of that group must have granted your user account the *Manage Object Promotion Import* right and you have Full Control permissions for the repository.
- You can only import objects to a local repository.
- When a datastore is promoted, its password is not included in the import or export operation, so the password field for the promoted datastore is left blank. This is intentional and provides a safeguard to prevent the unintended use of a development or QA datastore in a production environment.
  After a datastore promotion is completed, be sure to reconfigure the datastore under *Administration > Management > Datastore* in SAP Data Services Management Console. Set a password for the promoted datastore in its new location, and if needed, modify the location of the database it uses.
- For information about the general requirements for object promotion, see the related topics.

You can use object promotion to securely import objects into a production environment from the source directory for the import configuration.

1. In Data Services Management Console, select *Object Promotion*.
2. Select a target repository from which to import objects, then click *Next*.
   The *Import* page appears, with a list of exported objects that are available in the target repository. These entries contain information about objects that are associated with a corresponding export configuration, and contains information about where to find the exported objects.

   There are 23 entries listed on each page. You can include up to 23 of the entries listed on a given *Import* page in a single import operation. Click the page links to view any subsequent pages of available objects
3. To narrow the list to display only the objects you want to import, use the *Filter* choices to filter the displayed entries in the list or search for objects that completely or partially match the export configuration for the objects you want to import. The *Search* field supports wildcards.
4. Individually choose up to 23 entries displayed on the current page, or choose *Select All* to import all 23 objects displayed on the current page.
5. Click *Import* to start importing the selected objects.

The *Confirmation* page displays progress for the import, and the Status is automatically refreshed every five seconds. When the import is completed, the automatic refresh ends, the ending *Status* information is displayed (success or failure), and log information is stored in the repository under the *AL_IMPORT_HISTORY* table. To view the log information while you have the *Confirmation* page open, select *view log* under the *Status* column of the displayed report. Details about the job are on the *Trace* tab. If there were any issues with the export, messages are recorded to the *Error* tab.

A unique confirmation ID is assigned to each import attempt. The confirmation ID is used for a unique ID in the *AL_IMPORT_HISTORY* table. The confirmation ID contains the following information:

```
<Timestamp-in-milliseconds>-<six-digit random number>-<number of import attempts>
```

To convert the time stamp that is displayed in milliseconds to date format, copy the integers displayed next to *Confirmation#* up to the first dash, and paste them into any available conversion tool.

> i Note
>
> Make a note of the unique confirmation number displayed on the *Confirmation* page, and copy any information from the log that you want to preserve. You cannot return to this *Confirmation* page after you dismiss it. A copy of the log is also written to the repository under the *AL_IMPORT_HISTORY* table, so you can query the table to find the log information later.

If you imported a datastore, after it is imported you must reconfigure the datastore under *Administration > Management > Datastore* in SAP Data Services Management Console and assign it a password.

If you imported substitution parameters, if needed, you can update the paths for the substitution parameters after they are imported to their new location. To edit substitution parameters, you must either be an Administrator or your account is a member of the Data Services Administrator group, or either of these users can grant you the *Manage datastore and substitution param configurations*.

Related Information

Requirements for object promotion [page 132]
Setting up an import configuration [page 136]
Managing application rights for a group [page 51]

1.9 Integration with SAP and SAP Solution Manager

1.9.1 Integration overview

Data Services can be integrated into a number of SAP solutions to take advantage of their features. The System Landscape Directory and Solution Manager Diagnostics products help you manage, monitor, and maintain your Data Services deployment.

SAP System Landscape Directory (SLD)

The system landscape directory of SAP NetWeaver is the central source of system landscape information relevant for the management of your software life-cycle. By providing a directory comprising information about all installable software available from SAP and automatically updated data about systems already installed in a landscape, you get the foundation for tool support to plan software life-cycle tasks in your system landscape.

The SAP Data Services installation program registers the vendor and product names and versions with the SLD, as well as server and front-end component names, versions, and location.

Solution Manager Diagnostics (SMD)

The SMD component of SAP Solution Manager provides all functionality to centrally analyze and monitor a complete system landscape. Data Services can be monitored by the SMD server if an SMD Agent is installed. The SMD Agent gathers information for the SMD which can then be used for root cause analysis.

Data Services provides support for this performance monitoring through CA/Wily Introscope in Solution Manager Diagnostics through an integration with the NCS library, which is installed automatically with Data Services.

1.9.2 SLD and SAP Solution Manager integration checklist

The following table summarizes what components are required to enable SLD and SAP Solution Manager to provide support for Data Services.

Support for...	Required for SAP Data Services
SLD registration	• SAPHOSTAGENT must be installed to enable registration of Data Services servers. i Note The Data Services installer will automatically register servers if SAPHOSTAGENT is already installed. • Must create a `slddest.cfg.key` and `slddest.cfg` file for the SLDReg data supplier reporting on the back-end servers.
SMD integration	Must download and install SMD Agent (DIAGNOSTICS.AGENT) on all hosts of Data Services servers.
Performance instrumentation	• SMD Agent must be installed. • Introscope Agent must be configured to connect to Introsope Enterprise Manager. Use the Data Services Server Manager (Windows) or ServerConfig utility (UNIX) to configure the NCS options.

1.9.3 Managing System Landscape Directory registration

1.9.3.1 Registration of Data Services in the System Landscape

The System Landscape Directory (SLD) is a central repository of system landscape information that is relevant for the management of the software lifecycle. The SLD contains a description of the system landscape—the systems and software components that are currently installed.

SLD data suppliers (SLDReg) register the systems on the SLD server and keep the information up-to-date. Management and business applications access the information stored in the SLD to perform tasks in a collaborative computing environment.

SLDReg is installed when you install the SAPHOSTAGENT. Once SLDREG has been installed, you need to create a `slddest.cfg` and `slddest.cfg.key` file to enable it to connect to the SLD server.

The data supplier is provided for every installation of Data Services to report on the following components:

- Server components (job server, access server)
- Services deployed on the Business Intelligence Platform (RFC Server, View Data and Metadata Browsing Service, Administrator Service)
- Web applications deployed on an application server (Management Console)

> Note
>
> SAP NetWeaver has a built-in SLD-DS supplier that registers the NetWeaver application server as well as hosted web applications and services. This SLD-DS is relevant for Data Services deployments that are integrated within an SAP NetWeaver environment.

For information on how to configure the specific data supplier for WebSphere, see the *SAP Web Application Deployment Guide*.

During the installation of Data Services, information required for registering Data Services is stored in a configuration file. This file contains information used by the SLDReg to connect to the Data Services database.

1.9.3.2 To create a slddest.cfg.key file for the SLDReg

Before creating a `slddest.cfg.key` file for the SLD data supplier, you need to download and install the SAPHOSTAGENT.

> Note
>
> If you selected to add SLD during the Data Services installation, you do not need to create this file. If you are choosing to activate SLD after installing Data Services, follow this procedure.

The `slddest.cfg.key` file is required for SLD registration with the data supplier that reports on Data Services servers.

> Note
>
> This procedure creates both the `slddest.cfg` and the `slddest.cfg.key` file. Both of these files are required for SLD integration to work.

1. Open a command line console.
2. Navigate to the default SAPHOSTAGENT install path.
   - On Windows: `Program Files\SAP\hostctrl\exe`
   - On UNIX: `/usr/sap/hostctrl/exe`
3. Run the following command:

   ```
   sldreg -configure slddest.cfg
   ```
4. Enter the following configuration details:
   - User name
   - Password
   - Host
   - Port number
   - Specify to use HTTP

The `sldreg` tool will create the `slddest.cfg.key` file that will automatically be used by the data supplier to push information to the SLD server.

SLDReg needs to be running in the **`<LINK_DIR>`**`/sldreg` directory, or these files need to be manually copied to this directory for SLD integration to work.

1.9.3.3 When is SLD registration triggered?

The Data Services service invokes SLDReg (the data supplier for Data Services) to handle SLD registration.

1.9.4 Performance and availability monitoring

1.9.4.1 Solution Manager Diagnostics (SMD) overview

The Solution Manager Diagnostics (SMD) component of SAP Solution Manager provides all functionality to centrally analyze and monitor a complete system landscape. Data Services can be monitored by the SMD server if an SMD Agent is installed. The SMD Agent gathers information for the SMD which can then be used for root cause analysis. Information collected and sent to the SMD server includes back-end server configurations and the location of server log files.

Data Services provides support for performance and availability monitoring through CA Wily Introscope in Solution Manager Diagnostics through an integration with the NCS library, which is installed automatically with Data Services.

Components of SMD

- *SAP Solution Manager:* You must have Solution Manager 7.01 SP26 or later installed. For more information, see *https://service.sap.com/solutionmanager*.
- *SMD Agent:* A local agent (`DIAGNOSTICS.AGENT`) that collects and sends the information to the SMD server. This agent must be downloaded and installed on each Job Server that you want to monitor. The Data Services installation does not install this agent for you.

  Information on installing and configuring the agent is available at: *https://service.sap.com/diagnostics*.
- *CA Wily Introscope:* An application performance management framework. Introscope Enterprise Server is part of Solution Manager. There is no need to perform a separate installation. For more information, see *https://service.sap.com/diagnostics*
- *SAPOSCOL:* The SAP Operating System Collector provides operating system data to the SMD and Introscope.

All of these components are available for download from *http://service.sap.com/swdc*.

1.9.4.2 SMD agent guidelines

The SMD Agent is a local agent (`DIAGNOSTICS.AGENT`) that collects and sends the information to the SMD server.

All of these components are available for download from *http://service.sap.com/swdc*.

Guidelines for working with the SMD Agent

The following are provided as guidelines for using SMD agents to monitor Data Services:

- The installation order of the monitored system and agent is not critical. You can install the SMD Agent before or after installing and deploying Data Services.
- If the servers are deployed on a distributed system, you should install an SMD Agent on every machine hosting a server.

Related Information

http://service.sap.com/diagnostics

1.9.4.3 Configuring your system for SMD

There are a few settings and files to configure to get SMD working properly on your system.

- You must enable the feature in the Server Manager for each of the Job Servers for which you want to get performance metrics.
- If you have problems, you can edit the `ncs.conf` file. This file controls the information sent to the SMD Agent. Normally, you can keep the default settings. Descriptions of the options are included in the file. It is located in the **`<LINK_DIR>`**`\bin` directory of your Data Services installation.
- SMD Agent files. There are two files in your SMD Agent installation location: `SapAgentConfig.xml` and `IntroscopeSapAgent.profile`. Configuring these files is necessary to provide information to the Solution Manager server.

1.9.4.4 To enable performance instrumentation on Windows

Before you can monitor performance on a Job Server, you must enable it.

1. Open the Data Services Server Manager.
2. Click the *Native Component Supportability* tab.
3. Select *Enable instrumentation in NCS (Native Component Supportability) library*.

In most circumstances, you can leave the default settings for the rest of the options.

1.9.4.4.1 Server Manager: Native Component Supportability options

Option	Description
Enable instrumentation in NCS (Native Component Supportablility) library	Select to enable performance monitoring of the jobs run on this server.
Tracing level threshold	Indicates the tracing level that the instrumented code needs to go under to produce a trace: 0: Use the value from ncs.conf configuration file 1-5: No tracing (NONE) 6-10: Tracing major points (MAJOR) 11-15: Tracing minor points (MINOR) 16-20: Tracing fine details (FINE) >20: Max details (FINEST)
Execution interval	Indicates execution interval for CPU usage/process memory metrics to be sent to Wily Enterprise Manager in seconds. 0 means that the default NCS scheduler will be used.
Execution time offset	Indicates execution time offset with regard to the interval in seconds.

Option	Description
	For example, if the interval is 3600 (every one hour) and the offset is 1800 (every half an hour), the information will be sent to SMD agent at 3:30, 4:30, 5:30, and so on. If the interval is smaller than the NCS library scheduler interval defined in parameter "datasending_interval", the offset parameter will be bypassed.
Tracing level	This option is not currently used.
Application passport	This option is not currently used.

1.9.4.5 To enable performance instrumentation on UNIX and Linux

If you are running Data Services on a UNIX or Linux platform, you will need to edit the `DSConfig.txt` file to enable instrumentation.

1. Navigate to **`<DS_COMMON_DIR>`**`/conf`.
2. Open the `DSConfig.txt` file.
3. Set the Wily_instrumentation parameter to **`True`**.

The other parameters (found in the Engine section) can be left with the default values.

```
Wily_instrumentation=TRUE

Wily_instrumentation_Level_Threshold=0

Wily_instrumentation_Interval=0

Wily_instrumentation_Offset=

Wily_instrumentation_TraceLevel=3487

Wily_instrumentation_App_Passport=
```

1.9.4.6 Heartbeat monitoring

Availability monitoring (heartbeat) lets you use the SAP Solution Manager to check whether a component such as a Job Server or Access Server is up and running. You can also get information about real-time services for Access Servers.

From CA Wily Introscope under the *Status* node, you can view each monitored Job Server or Access Server's status. For heartbeat, a value of 1 indicates the server is running; 0 indicates it is not.

In addition, you can view an Access Server's real-time service status. The status indicators are:

0	not running
1	starting

2	started
3	shutting down
4	warning
5	error
9	disabled

1.9.4.7 Alert monitoring

Alerts let you view critical errors in the SAP Solution Manager. From Data Services, Job Servers send alerts when a job fails.

From CA Wily Introscope under the *Status* node, you can view each monitored Job Server's status. For alerts, a value of 1 indicates a job has failed in that Job Server's repository.

1.10 Command line administration

1.10.1 Command lines overview

This section lists the command-line options that control the behavior of each Data Services component.

Throughout this section, values provided in square brackets [ ] are optional.

i Note

The following tables list the supported command-line options. Data Services components use a number of internal options that are not listed in these tables. These internal options should not be modified.

1.10.2 License Manager

License Manager includes a command-line mode that you can use if you don't want to use the graphical interface, or need to script License Manager operations.

Syntax

```
LicenseManager [-v | -a <keycode> | -r <keycode> [-l <location>]]
```

Parameter	Description

-v or **--view**	Displays the stored product activation keycodes in a format similar to the License Manager graphical interface. For example:
	`------ Registered Keycodes ------`
	` EIM Titan`
	`  Suite 12.0`
	`   Data Services XI 3.1 Premium`
	`    Keycode: 00000-0000000-0000000-0000`
	`    Trial Option: Yes`
	`    Expired: No`
	`    Days Remaining: 54`

-a or **--add <keycode>**	Adds the specified license keycode, and displays the stored keycodes in a format similar to the License Manager graphical interface. Returns status messages for the following conditions: - An internal error occurred. - Successfully added the keycode. - Successfully added the keycode and replaced a trial version. - Keycode not added because it is invalid. - Keycode not added because it is a duplicate.
-r or **--remove <keycode> [-l <location>]**	Removes the specified product activation keycode, and displays the stored keycodes in a format similar to the License Manager graphical interface. If **<location>** is specified, the removal is restricted to that node. Returns status messages for the following conditions: - An internal error occurred. - Removed one keycode. - Removed multiple keycodes. - Keycode not removed because it is invalid. - Keycode not removed because it was not found.

1.10.3 Connection Manager (Unix)

The Connection Manager (DSConnectionManager) is a graphical interface used to configure ODBC databases and ODBC drivers that you want to use for Data Services repositories, sources and targets after installation on Unix platforms. The Connection Manager includes a command-line mode that you can use if you do not want to use the graphical interface, or need to troubleshoot errors.

To use `DSConnectionManager.sh` from the command line, use the `-c` parameter which must be the first parameter.

If an error occurs when using the Connection Manager, use the `-d` option to show details in the log

For example:

```
$LINK_DIR/bin/DSConnectionManager.sh -c -d
```

1.10.4 Repository Manager (Windows)

You can use RepoManBatch.exe to create or update repositories from the command line on Windows platforms. By default, RepoManBatch.exe is installed to the **<LINK_DIR>**\bin directory.

Specify parameters using a command prompt.

```
C:\Program Files\SAP BusinessObjects\Data Services\bin>RepoManBatch.exe

Usage:
     -U<User>          : Repository login user
     -P<Password>      : Repository login password
     -s                : Use Server name based connection
     -S<Server>        : Repository server name
     -p<PortNo>        : Repository server port number
     -N<DatabaseType> : Repository database type
     -Q<Database>      : Repository database
     -V<Database_version>    : Repository database server version
     -g                : Repository using Windows Authentication (Microsoft
                            SQL Server only)
     -t<Type>          : Repository type: local, central, profiler
     -c                : Repository create
     -u                : Repository upgrade
     -v                : Repository version
     -d                : Show details
     -a                : Repository security

C:\Program Files\SAP BusinessObjects\Data Services\bin>
```

For example:

```
RepoManBatch -Usa -P -NMicroSoft_SQL_Server -SServer -QJake -c -tcentral -d
```

or

```
RepoManBatch -UJake -PJake -NOracle -Sdbsvr -v
```

Usage:

Flag	Description
-U	Repository login user This parameter is required for all database types.
-P	Repository login password This parameter is required for all database types.
-s	Specify this parameter to use a server name (also known as DSN-less or TNS-less) connection to the repository. If you specify this parameter, you must specify the -p and -V parameters.
-S	Repository server name: • For DB2: data source • For MySQL: ODBC data source name • For Microsoft SQL Server: database server name • For Oracle: database connection name • For SAP HANA: ODBC data source name • For SAP Sybase SQL Anywhere: ODBC data source name • For SAP Sybase ASE: server

Flag	Description
	This parameter is required for all database types.
`-p`	Repository database port number This parameter is required if you specified `-s` for a server name connection.
`-N`	Repository database type: ● `DB2` ● `HANA` ● `Microsoft_SQL_Server` ● `MySQL` ● `Oracle` ● `SQL_Anywhere` ● `Sybase` This parameter is required for all database types.
`-Q`	Repository database name This parameter is required only for Microsoft SQL Server and Sybase ASE.
`-V`	Repository database version This parameter is required if you specified `-s` for a server name connection.
`-g`	Specify this parameter to use Windows authentication to connect to this repository (Microsoft SQL Server only).
`-t`	Repository type: ● `local` ● `central` ● `profiler`
`-c`	Create repository
`-u`	Upgrade repository
`-v`	Get repository version
`-d`	Show details
`-a`	Central repository security

1.10.5 Repository Manager (Unix)

You can use the executable called `repoman` to create or update repositories from the command line on Unix platforms. By default, `repoman` is installed to the **`<LINK_DIR>`**`/bin` directory.

Specify parameters using a command prompt.

```
$ ./repoman
```

```
Usage:
     -U<User>         : Repository login user
     -P<Password>     : Repository login password
     -S<Server>       : Repository server name
     -N<DatabaseType> : Repository database type: SQL_Anywhere, Sybase, MySQL, HANA,
                           DB2, Oracle
     -Q<Database>     : Repository database
     -s               : Connect Repository database by DSN-less (ODBC) or
                           TNS-less for Oracle
     -V<databaseVersion> : Repository database version (only available
                           with -s):
                              MYSQL 5.0, MYSQL 5.1 (default)
                              HANA 1.X (default)
                              DB2 UDB 9.X
                              ORACLE 11G, ORACLE 10G

                                  SQL Anywhere 12.X
     -p<port>         : Repository database port
     -t<Type>         : Repository type: local, central,
                           profiler
     -b               : Check database connectivity
     -c               : Repository create
     -u               : Repository upgrade
     -v               : Repository version
     -d               : Show details
     -a               : Repository security
     -l               : Create log file
     -z               : Create error file
                           (local, central, profiler modes)
```

For example:

```
./repoman -Usa -P -NDB2 -SServer -QJake -c -tcentral -d
```

or

```
./repoman -UJake -PJake -NOracle -Sdbsvr -v
```

Usage:

Flag	Description
`-U`	Repository login user This parameter is required for all database types.
`-P`	Repository login password This parameter is required for all database types.
`-s`	Specify this parameter to use a server name (also known as DSN-less or TNS-less) connection to the repository. If you specify this parameter, you must specify the `-p` and `-V` parameters.
`-S`	Repository server name: • For DB2: data source • For MySQL: ODBC data source name • For Oracle: TNSNAME defined in `tnsnames.ora` • For SAP HANA: ODBC data source name • For SAP Sybase SQL Anywhere: ODBC data source name • For SAP Sybase ASE: server This parameter is required for all database types.

Flag	Description
`-p`	Repository database port number This parameter is required if you specified `-s` for a server name connection.
`-N`	Repository database type: • `DB2` • `HANA` • `MySQL` • `Oracle` • `SQL_Anywhere` • `Sybase` This parameter is required for all database types.
`-Q`	Repository database name This parameter is required only for Sybase ASE.
`-V`	Repository database version This parameter is required if you specified `-s` for a server name connection.
`-t`	Repository type: • `local` • `central` • `profiler`
`-c`	Operation mode: Creates repository
`-u`	Operation mode: Upgrades repository
`-v`	Operation mode: Gets repository version
`-d`	Operation mode: Shows details
`-a`	Central repository security
`-o`	Overwrite existing repository

1.10.6 Server Manager (Windows)

The Server Manager (`AWServerConfig.exe`) is used to create, edit, or delete Job Servers and Access Servers after installation on Windows platforms. In addition to the default graphical user interface, `AWServerConfig.exe` also supports command-line parameters for several tasks:

- Adding a Job Server
- Adding an Access Server
- Adding run-time resources

Note

On Windows platforms, there is no command-line option to start or stop the Data Services service using `AWServerConfig.exe` because it is installed as a Windows service. The Data Services service can be started and stopped using the standard `net` command.

Example

Start Data Services services

```
net start "SAP Data Services"
```

Example

Stop Data Services services

```
net stop "SAP Data Services"
```

1.10.6.1 To add an Access Server

To use `AWServerConfig.exe` to add an Access Server from the command line, use the `-n` parameter, along with additional Access Server-specific parameters. `-n` must be the first argument.

Access Server parameters

Parameter	Description
`-R`**`<access_server_dir>`**	Specifies the directory path for the Access Server. Replace **`<access_server_dir>`** with the Access Server directory path.
`-A`**`<port>`**	Specifies the port assigned to the Access Server. Replace **`<port>`** with the desired port number. The port number may have a value between 1024 and 49151, and must be unique and not in use.
`-E`	Indicates that the Access Server should be enabled. If not specified, the Access Server is created but not enabled.
`-T`**`<param>`**	Specifies a parameter for the Access Server. Replace **`<param>`** with the desired parameter.

Example

Create and enable an Access Server on port 4000

```
AWServerConfig.exe -n -RC:\DataServices\AccessServer -A4000 -E
```

1.10.6.2 To add a Job Server

To use `AWServerConfig.exe` to add a Job Server from the command line, use the `-n` parameter, along with additional Job Server-specific parameters. `-n` must be the first parameter.

Job Server parameters

Parameter	Description
`-J`**`<server_name>`**	Specifies the name of the Job Server. Replace **`<server_name>`** with the desired name for the Job Server. The specified name may not contain `@@` and must be unique.
`-P`**`<port_number>`**	Specifies the listening port for the Job Server. Replace **`<port_number>`** with the desired port number. The port number may have a value between 1024 and 49151, and must be unique and not in use.
`-a`	Indicates that the Job Server will manage an adapter. If not specified, the new Job Server will not manage adapters.
`-B`**`<broker_port>`**	Specifies the adapter manager port. Replace **`<broker_port>`** with the desired port number. The port number may have a value between 1024 and 49151, and must be unique and not in use.
`-s`	Indicates that SNMP is enabled for the Job Server. If not specified, SNMP is disabled.
`-d`	Indicates that this is the default repository for the Job Server.
`-U`**`<username>`**	Specifies the username used to connect to the repository. Replace **`<username>`** with the repository username.
`-W`**`<password>`**	Specifies the password used to connect to the repository. Replace **`<password>`** with the repository password.
`-N`**`<db_type>`**	Specifies the type of database used for the repository. Replace **`<db_type>`** with a valid value: ● `DB2` ● `Microsoft_SQL_Server` ● `MySQL` ● `Oracle` ● `SQL_Anywhere` ● `Sybase`

Parameter	Description
-S**<server_name>**	Specifies the database service name or server name used to connect to the repository. Replace **<server_name>** with the appropriate information for the database type: ● For Oracle, the database service name as specified in tnsnames.ora. ● For DB2, the database instance name. ● For Microsoft SQL Server, the database server name. ● For Sybase, the database server name. ● For MySQL, the database source name as specified in the system DSN. ● For SAP Sybase SQL Anywhere, the database server name.
-Q**<database_name>**	Specifies the database name for the repository. Replace **<database_name>** with the name of the repository database. Note This parameter is required only for repositories on Sybase and Microsoft SQL Server.
-g	Indicates that Windows authentication will be used for the connection to the repository. Note This parameter is applicable only for repositories on Microsoft SQL Server.

Example

Add a Job Server with an Oracle repository

```
AWServerConfig.exe -n -JNewJobServer -P3500 -Uuser -Wpass -NOracle -SORCLPROD
```

1.10.6.3 To add run-time resources

To use `AWServerConfig.exe` to add run-time resources from the command line, use the `-n` parameter, along with additional run-time resource-specific parameters. `-n` must be the first parameter.

Run-time resource parameters

Parameter	Description
-C**<cache_dir>**	Specifies the directory for the pageable cache. Replace **<cache_dir>** with the desired directory.
-PF**<from_port>**	Specifies the starting port number. Replace **<from_port>** with the desired port number. The port number may have a value between 1025 and 32766, and must be unique and not in use.
-PT**<to_port>**	Specifies the ending port number. Replace **<to_port>** with the desired port number. The port number may have a value between 1026 and 32767, and must be unique and not in use. The ending port value must be greater than the starting port value.

Example

Add a pageable cache resource on ports 2000-2550

```
AWServerConfig.exe -n -C"%LINK_DIR%\log\Pcache" -PF2000 -PT2550
```

1.10.7 Server Manager (Unix)

The Server Manager (`svrcfg`) is used to create, edit, or delete Job Servers and Access Servers after installation on Unix platforms. In addition to the console-based interface, `svrcfg` also supports command-line parameters for several tasks:

- Adding a Job Server
- Adding an Access Server
- Adding run-time resources
- Starting Data Services services
- Stopping Data Services services

Common parameters

`svrcfg` supports one common parameter for all operations. Other available parameters depend on the operation.

Parameter	Description
-T**<task>**	Specifies the configuration task to perform. Available values for **<task>** include:

Parameter	Description
	• `JS` - Add a Job Server • `AS` - Add an Access Server • `R` - Add run-time resources • `START` - Start services • `STOP` - Stop services

ℹ Note

When starting or stopping the Data Services services, `svrcfg` requires no additional parameters.

Example

Start Data Services services

```
svrcfg -TSTART
```

Example

Stop Data Services services

```
svrcfg -TSTOP
```

Output

When using `svrcfg` to perform tasks from the command line, output is directed to the console (or `stdout`). The last line of the output indicates whether the task execution succeeded or failed. Possible statuses include:

- `Success`
- `Failure`

1.10.7.1 To add an Access Server

To use `svrcfg` to add an Access Server from the command line, use the `-TAS` parameter, along with additional Access Server-specific parameters.

Access Server parameters

Parameter	Description
-A**<path>**	Specifies the path for the access server. Replace **<path>** with the desired path.
-O**<port>**	Specifies the port assigned to the Access Server. Replace **<port>** with the desired port number. The port number may have a value between 1024 and 49151, and must be unique and not in use.
-R**<param>**	Specifies a parameter for the Access Server. Replace **<param>** with the desired parameter.
-E	Indicates that the Access Server should be enabled. If not specified, the Access Server is created but not enabled.

Example

Create and enable an Access Server on port 4000

```
svrcfg -TAS -A/home/bods/AStest -O4000 -E
```

1.10.7.2 To add a Job Server

To use `svrcfg` to add a Job Server from the command line, use the `-TJS` parameter, along with additional Job Server-specific parameters.

Job Server parameters

Parameter	Description
-J**<server_name>**	Specifies the name of the Job Server. Replace **<server_name>** with the desired name for the Job Server. The specified name may not contain `@@` and must be unique.
-p**<port_number>**	Specifies the listening port for the Job Server. Replace **<port_number>** with the desired port number. The port number may have a value between 1024 and 49151, and must be unique and not in use.
-a	Indicates that the Job Server will manage an adapter. If not specified, the new Job Server will not manage adapters.

Parameter	Description
`-b`**`<broker_port>`**	Specifies the adapter manager port. Replace **`<broker_port>`** with the desired port number. The port number may have a value between 1024 and 49151, and must be unique and not in use.
`-e`	Indicates that SNMP is enabled for the Job Server. If not specified, SNMP is disabled.
`-D`**`<db_type>`**	Specifies the type of database used for the repository. Replace **`<db_type>`** with a valid value: ● `DB2` ● `HANA` ● `MySQL` ● `Oracle` ● `SQL_Anywhere` ● `SYBASE`
`-C`**`<connect_string>`**	Specifies the connection string to use to connect to the repository. Replace **`<connect_string>`** with appropriate information for the database type: ● For DB2, the database instance name. ● For MySQL, the data source name as specified in the `odbc.ini` file referenced by `$ODBCINI`. ● For Oracle, the service name as specified in `tnsnames.ora`. ● For SAP HANA, the data source name as specified in the `odbc.ini` file referenced by `$ODBCINI`. ● For SAP Sybase SQL Anywhere, the database server name. ● For Sybase, the database server name.
`-d`**`<database>`**	Specifies the database name for the repository. Replace **`<database>`** with the name of the repository database. ℹ Note This parameter is required only for repositories on Sybase.
`-U`**`<username>`**	Specifies the username used to connect to the repository. Replace **`<username>`** with the repository username.
`-P`**`<password>`**	Specifies the password used to connect to the repository. Replace **`<password>`** with the repository password.

Example

Add a Job Server with an Oracle repository

```
svrcfg -TJS -JJobServer_1 -p3500 -DOracle -CORCL -Uuser -Ppassword
```

1.10.7.3 To add run-time resources

To use `svrcfg` to add run-time resources from the command line, use the `-TR` parameter, along with additional run-time resource-specific parameters.

Run-time resource parameters

Parameter	Description
`-i<cache_dir>`	Specifies the directory for the pageable cache. Replace `<cache_dir>` with the desired directory.
`-t<port>`	Specifies the starting port number. Replace `<port>` with the desired port number. The port number may have a value between 1025 and 32766, and must be unique and not in use.
`-n<port>`	Specifies the ending port number. Replace `<port>` with the desired port number. The port number may have a value between 1026 and 32767, and must be unique and not in use. The ending port value must be greater than the starting port value.

Example

Add a pageable cache resource on ports 2000-3000

```
svrcfg -TR -i$LINK_DIR\Log\Cache2 -t2000 -n3000
```

1.10.8 Password encryption

You can use `al_encrypt` to encrypt a password by using either an encryption key or a passphrase.

Additionally, you can use `al_encrypt` to return the base64 encoding of any text. This may be useful if you need to modify a command line that contains global variable or substitution variable data, which must be encoded in base64 form.

By default, `al_encrypt` is installed to the **`<LINK_DIR>`**`/bin` directory.

Syntax

```
al_encrypt -e <plain_password> [-k <key string> | -p <passphrase>]
```

```
al_encrypt "<text to encode>"
```

Parameter	Description
`-e <password>`	Specifies the plain-text password to encrypt.
`-k <key string>`	Specifies the encryption key to use to encrypt the password.
`-p <passphrase>`	Specifies the passphrase to use to encrypt the password.
`<text to encode>`	When you run `al_encrypt` with no parameters, it returns the base64 encoding of any following optionally-quoted text.

Example

Encrypt a password using a passphrase

```
al_encrypt -e mypassword -p thepassphrase >
+0100000000120303000803E83F55088B0C987CD715006C02938825530E8691DFD9DDB4198AFFC5C194C
D8CE6D338FDE470E2
```

Example

Encode text using base64 encoding

```
al_encrypt "encode this as base64" > ZW5jb2RlIHRoaXMgYXMgYmFzZTY0
```

1.10.9 al_engine

`al_engine` is a core Data Services process. It is responsible for executing jobs, importing and exporting repository objects, and so on.

Common options

`al_engine` supports options that are common to many different operations.

Parameter	Description
`-U<Username>`	Specifies the username used to log into the repository.
`-P<Password>`	Specifies the password used to log into the repository.
`-S<ServerName>`	Specifies the repository server name. For a DSN connection to a DB2, SAP HANA SAP Sybase, or SAP Sybase SQL Anywhere repository, use ODBC connection name.

Parameter	Description
-N**<DatabaseType>**	Specifies the repository database type Acceptable values include: ● Oracle ● Microsoft_SQL_Server ● DB2 ● MySQL ● Sybase ● HANA ● SQL_Anywhere
-Q**<DatabaseName_or_SID>**	Specifies the repository database name or SID (for Oracle). For a DSN-less connection to a DB2, SAP Sybase, or SAP Sybase SQL Anywhere repository, use database name.
-Kserver	Specify this parameter to use a server name (also known as DSN-less or TNS-less) connection to the repository. If you specify this parameter, you must specify the -Kport and -Kversion parameters.
-Kport**<PortNumber>**	Repository port number for server name connection. This parameter is required if you specified -Kserver for a server name connection.
-Kversion**<VersionNumber>**	Repository database server version for server name connection. This parameter is required if you specified -Kserver for a server name connection. For example, -Kversion"MySQL 5.1"
-g	Specifies Windows Authentication as the repository connection type. This parameter is valid only for repositories on Microsoft SQL Server.
-v	Returns the version number of the Data Services engine.

1.10.9.1 Export and import options

al_engine supports options that are used only for export and import operations. al_engine can import and export repository information in two formats: XML and ATL, the software's internal scripting language.

Parameter	Description
-X	Exports the entire repository in ATL format to repo_export.atl.
-XKserver	Exports repository server name connection (for MySQL, SAP HANA, ORACLE, and DB2).
-XKport**<PortNumber>**	Exports repository port number for server name connection. This parameter must be used with -XKserver parameter.

Parameter	Description
-XKversion**<VersionNumber>**	Exports repository database server version for server name connection. This parameter must be used with the –XKserver parameter. For example, -XKversion"MySQL 5.1"
-Xp@**<ObjectType>**@**<FileName>**	Exports all repository objects of the specified type to the specified file in ATL format.
-Xp@**<ObjectType>**@**<FileName>**@**<ObjectName>**	Exports the specified repository object to the specified file in ATL format.
–Xp@**<ObjectType>**@**<FileName>**@**<ObjectName>**@DE	Exports the specified repository object and its dependents to the specified file in ATL format, including datastore information.
-Xp@**<ObjectType>**@**<FileName>**@**<ObjectName>**@D	Exports the specified repository object and its dependents to the specified file in ATL format, excluding datastore information.
-XX[L]	Exports the entire repository in XML format to export.xml.
-XX[L]@**<ObjectType>**@**<FileName>**	Exports all repository objects of the specified type to the specified file in XML format.
–XX[L]@**<ObjectType>**@**<FileName>**@**<ObjectName>**	Exports the specified repository object to the specified file in XML format.
–XX[L]@**<ObjectType>**@**<FileName>**@**<ObjectName>**@DE	Exports the specified repository object and its dependents to the specified file in XML format, including datastore information.
–XX[L]@**<ObjectType>**@**<FileName>**@**<ObjectName>**@D	Exports the specified repository object and its dependents to the specified file in XML format, excluding datastore information.
-f**<filename.atl>**	Imports information from **<filename.atl>** into the repository.
-XI**<filename.xml>**	Imports information from **<filename.xml>** into the repository.
-passphrase**<passphrase>**	Specifies a plain-text passphrase to use to encrypt any passwords when exporting objects or decrypt any passwords when importing objects.
-epassphrase**<passphrase>**	Specifies a base64-encoded passphrase to use to encrypt any passwords when exporting objects or decrypt any passwords when importing objects. This parameter can be used to use a passphrase that contains special characters. Note You must transcode the passphrase to the UTF8 character set before encoding it into base64.

Note

For all –XX parameters, the optional addition [L] specifies a lean XML format for export. The lean XML format excludes all non-executable elements from the exported XML to improve readability. For example, the exact arrangement of transforms within a data flow in the Designer workspace area would not be maintained. When imported back into the software, the transforms would be arranged automatically.

Note

When you export objects, you must specify a passphrase with either the -passphrase parameter or the –epassphrase parameter. When you import objects, the passphrase is optional. However, if you do not specify a passphrase, or the specified passphrase is incorrect, any encrypted passwords in the imported objects will be removed.

Available object type codes

Code	Object type
P	Projects
J	Jobs
W	work flows
D	data flows
T	Idocs
F	User-defined file formats
X	XML and DTD message formats
S	Datastores
C	Custom functions
B	COBOL Copybooks
E	Excel workbooks
p	System profiles
v	Substitution parameter configurations
K	SDK-type transform configurations

Example

Export all data flows in lean XML format

```
al_engine -Uuser -Ppassword -Slocalhost -NMySQL -QTheRepository -
XXL@D@exported_dataflows.xml -passphraseMypassphrase
```

2 Designer Guide

2.1 Introduction

2.1.1 Welcome to SAP Data Services

2.1.1.1 Welcome

SAP Data Services delivers a single enterprise-class solution for data integration, data quality, data profiling, and text data processing that allows you to integrate, transform, improve, and deliver trusted data to critical business processes. It provides one development UI, metadata repository, data connectivity layer, run-time environment, and management console—enabling IT organizations to lower total cost of ownership and accelerate time to value. With SAP Data Services, IT organizations can maximize operational efficiency with a single solution to improve data quality and gain access to heterogeneous sources and applications.

2.1.1.2 Documentation set for SAP Data Services

You should become familiar with all the pieces of documentation that relate to your SAP Data Services product. The latest Data Services documentation can be found on the *SAP Help Portal*.

Document	What this document provides
Adapter SDK Guide	Information about installing, configuring, and running the Data Services Adapter SDK
Administrator Guide	Information about administrative tasks such as monitoring, lifecycle management, security, and so on.
Customer Issues Fixed	Information about customer issues fixed in this release. Note In some releases, this information is displayed the Release Notes.
Designer Guide	Information about how to use Data Services Designer.
Documentation Map	Information about available Data Services books, languages, and locations.
Installation Guide for Windows	Information about and procedures for installing Data Services in a Windows environment.
Installation Guide for UNIX	Information about and procedures for installing Data Services in a UNIX environment.
Integrator Guide	Information for third-party developers to access Data Services functionality using web services and APIs.

Document	What this document provides
Master Guide	Information about the application, its components and scenarios for planning and designing your system landscape. Information about SAP Information Steward is also provided in this guide.
Management Console Guide	Information about how to use Data Services Administrator and Data Services Metadata Reports.
Performance Optimization Guide	Information about how to improve the performance of Data Services.
Reference Guide	Detailed reference material for Data Services Designer.
Release Notes	Important information you need before installing and deploying this version of Data Services.
Technical Manuals	A compiled, searchable, “master” PDF of core Data Services books: ● *Administrator Guide* ● *Designer Guide* ● *Reference Guide* ● *Management Console Guide* ● *Performance Optimization Guide* ● *Integrator Guide* ● *Supplement for J.D. Edwards* ● *Supplement for Oracle Applications* ● *Supplement for PeopleSoft* ● *Supplement for Salesforce.com* ● *Supplement for Siebel* ● *Supplement for SAP* ● *Workbench Guide*
Text Data Processing Extraction Customization Guide	Information about building dictionaries and extraction rules to create your own extraction patterns to use with Text Data Processing transforms.
Text Data Processing Language Reference Guide	Information about the linguistic analysis and extraction processing features that the Text Data Processing component provides, as well as a reference section for each language supported.
Tutorial	A step-by-step introduction to using Data Services.
Upgrade Guide	Information to help you upgrade from previous releases of Data Services and release-specific product behavior changes from earlier versions of Data Services to the latest release.
What's New	Highlights of new key features in this SAP Data Services release. This document is not updated for support package or patch releases.
Workbench Guide	Provides users with information about how to use the Workbench to migrate data and database schema information between different database systems.

In addition, you may need to refer to several Supplemental Guides.

Document	What this document provides
Supplement for SAP	Information about interfaces between Data Services, SAP Applications, SAP Master Data Services, SAP NetWeaver BW, and SAP Master Data Services.
Supplement for SuccessFactors	Information about interfaces between Data Services and SuccessFactors.
Supplement for Salesforce.com	Information about how to install, configure, and use the SAP Data Services Salesforce.com Adapter Interface.
Supplement for J.D. Edwards	Information about interfaces between Data Services and J.D. Edwards World and J.D. Edwards OneWorld.
Supplement for Oracle Applications	Information about the interface between Data Services and Oracle Applications.
Supplement for PeopleSoft	Information about interfaces between Data Servicesand PeopleSoft.
Supplement for Siebel	Information about the interface between Data Services and Siebel.

We also include these manuals for information about SAP BusinessObjects Information platform services.

Document	What this document provides
Information platform services Administrator Guide	Information for administrators who are responsible for configuring, managing, and maintaining an Information platform services installation.
Information platform services Installation Guide for UNIX	Installation procedures for SAP BusinessObjects Information platform services on a UNIX environment.
Information platform services Installation Guide for Windows	Installation procedures for SAP BusinessObjects Information platform services on a Windows environment.

2.1.1.3 Accessing documentation

You can access the complete documentation set for SAP Data Services in several places.

2.1.1.3.1 Accessing documentation on Windows

After you install SAP Data Services, you can access the documentation from the Start menu.

1. Choose *Start* > *Programs* > *SAP Data Services 4.2* > *Data Services Documentation* > *All Guides* .
2. Click the appropriate shortcut for the document that you want to view.

2.1.1.3.2 Accessing documentation on UNIX

After you install SAP Data Services, you can access the documentation by going to the directory where the printable PDF files were installed.

1. Go to **<LINK_DIR>**`/doc/book/en/`.
2. Using Adobe Reader, open the PDF file of the document that you want to view.

2.1.1.3.3 Accessing documentation from the Web

You can access the complete documentation set for SAP Data Services from the SAP Business Users Support site.

To do this, go to *http://help.sap.com/bods*.

You can view the PDFs online or save them to your computer.

2.1.1.4 SAP information resources

A global network of SAP technology experts provides customer support, education, and consulting to ensure maximum information management benefit to your business.

Useful addresses at a glance:

Address	Content
Customer Support, Consulting, and Education services *http://service.sap.com/*	Information about SAP support programs, as well as links to technical articles, downloads, and online forums. Consulting services can provide you with information about how SAP can help maximize your information management investment. Education services can provide information about training options and modules. From traditional classroom learning to targeted e-learning seminars, SAP can offer a training package to suit your learning needs and preferred learning style.
Product documentation *http://help.sap.com/bods/*	SAP product documentation.
Supported Platforms (Product Availability Matrix) *https://service.sap.com/PAM*	Get information about supported platforms for SAP Data Services. Use the search function to search for Data Services. Click the link for the version of Data Services you are searching for.
SAP Data Services Community Network *http://scn.sap.com/community/data-services*	Get online and timely information about SAP Data Services, including forums, tips and tricks, additional downloads, samples, and much more. All content is to and from the community, so feel free to join in and contact us if you have a submission.
Blueprints *http://scn.sap.com/docs/DOC-8820*	Blueprints for you to download and modify to fit your needs. Each blueprint contains the necessary SAP Data Services project, jobs, data flows, file formats, sample data, template

Address	Content
	tables, and custom functions to run the data flows in your environment with only a few modifications.
SAPTerm *https://portal.wdf.sap.corp/go/sapterm*	SAP's terminology database, the central repository for defining and standardizing the use of specialist terms.

2.1.2 Overview of this guide

Welcome to the *Designer Guide*. The Data Services Designer provides a graphical user interface (GUI) development environment in which you define data application logic to extract, transform, and load data from databases and applications into a data warehouse used for analytic and on-demand queries. You can also use the Designer to define logical paths for processing message-based queries and transactions from Web-based, front-office, and back-office applications.

2.1.2.1 About this guide

The guide contains two kinds of information:

- Conceptual information that helps you understand the Data Services Designer and how it works
- Procedural information that explains in a step-by-step manner how to accomplish a task

You will find this guide most useful:

- While you are learning about the product
- While you are performing tasks in the design and early testing phase of your data-movement projects
- As a general source of information during any phase of your projects

2.1.2.2 Who should read this guide

This and other Data Services product documentation assumes the following:

- You are an application developer, consultant, or database administrator working on data extraction, data warehousing, data integration, or data quality.
- You understand your source data systems, RDBMS, business intelligence, and messaging concepts.
- You understand your organization's data needs.
- You are familiar with SQL (Structured Query Language).
- If you are interested in using this product to design real-time processing, you should be familiar with:
  - DTD and XML Schema formats for XML files
  - Publishing Web Services (WSDL, REST, HTTP, and SOAP protocols, etc.)
- You are familiar Data Services installation environments—Microsoft Windows or UNIX.

2.2 Logging into the Designer

You must have access to a local repository to log into the software. Typically, you create a repository during installation. However, you can create a repository at any time using the Repository Manager, and configure access rights within the Central Management Server.

Additionally, each repository must be associated with at least one Job Server before you can run repository jobs from within the Designer. Typically, you define a Job Server and associate it with a repository during installation. However, you can define or edit Job Servers or the links between repositories and Job Servers at any time using the Server Manager.

When you log in to the Designer, you must log in as a user defined in the Central Management Server (CMS).

1. Enter your user credentials for the CMS.

Option	Description
System	Specify the server name and optionally the port for the CMS.
User name	Specify the user name to use to log into CMS.
Password	Specify the password to use to log into the CMS.
Authentication	Specify the authentication type used by the CMS.

2. Click *Log on*.
   The software attempts to connect to the CMS using the specified information. When you log in successfully, the list of local repositories that are available to you is displayed.
3. Select the repository you want to use.
4. Click *OK* to log in using the selected repository.
   When you click *OK*, you are prompted to enter the password for the Data Services repository. This default behavior can be changed by adding the necessary rights to the repository in the CMC. See the *Administrator Guide* for more information.

Related Information

Administrator Guide: Password protection for a Data Services repository [page 45]
Administrator Guide: Managing security settings for repositories in the CMC [page 53]

2.2.1 Version restrictions

Your repository version must be associated with the same major release as the Designer and must be less than or equal to the version of the Designer.

During login, the software alerts you if there is a mismatch between your Designer version and your repository version.

After you log in, you can view the software and repository versions by selecting *Help* > *About Data Services*.

Some features in the current release of the Designer might not be supported if you are not logged in to the latest version of the repository.

2.2.2 Resetting users

Occasionally, more than one person may attempt to log in to a single repository. If this happens, the Reset Users window appears, listing the users and the time they logged in to the repository.

From this window, you have several options. You can:

- *Reset Users* to clear the users in the repository and set yourself as the currently logged in user.
- *Continue* to log in to the system regardless of who else might be connected.
- *Exit* to terminate the login attempt and close the session.

Note

Only use *Reset Users* or *Continue* if you know that you are the only user connected to the repository. Subsequent changes could corrupt the repository.

2.3 Designer User Interface

This section provides basic information about the Designer's graphical user interface.

2.3.1 Objects

All "entities" you define, edit, or work with in Designer are called objects. The local object library shows objects such as source and target metadata, system functions, projects, and jobs.

Objects are hierarchical and consist of:

- Options, which control the operation of objects. For example, in a datastore, the name of the database to which you connect is an option for the datastore object.
- Properties, which document the object. For example, the name of the object and the date it was created are properties. Properties describe an object, but do not affect its operation.

The software has two types of objects: Reusable and single-use. The object type affects how you define and retrieve the object.

2.3.1.1 Reusable objects

You can reuse and replicate most objects defined in the software.

After you define and save a reusable object, the software stores the definition in the local repository. You can then reuse the definition as often as necessary by creating calls to the definition. Access reusable objects through the local object library.

A reusable object has a single definition; all calls to the object refer to that definition. If you change the definition of the object in one place, you are changing the object in all other places in which it appears.

A data flow, for example, is a reusable object. Multiple jobs, like a weekly load job and a daily load job, can call the same data flow. If the data flow changes, both jobs use the new version of the data flow.

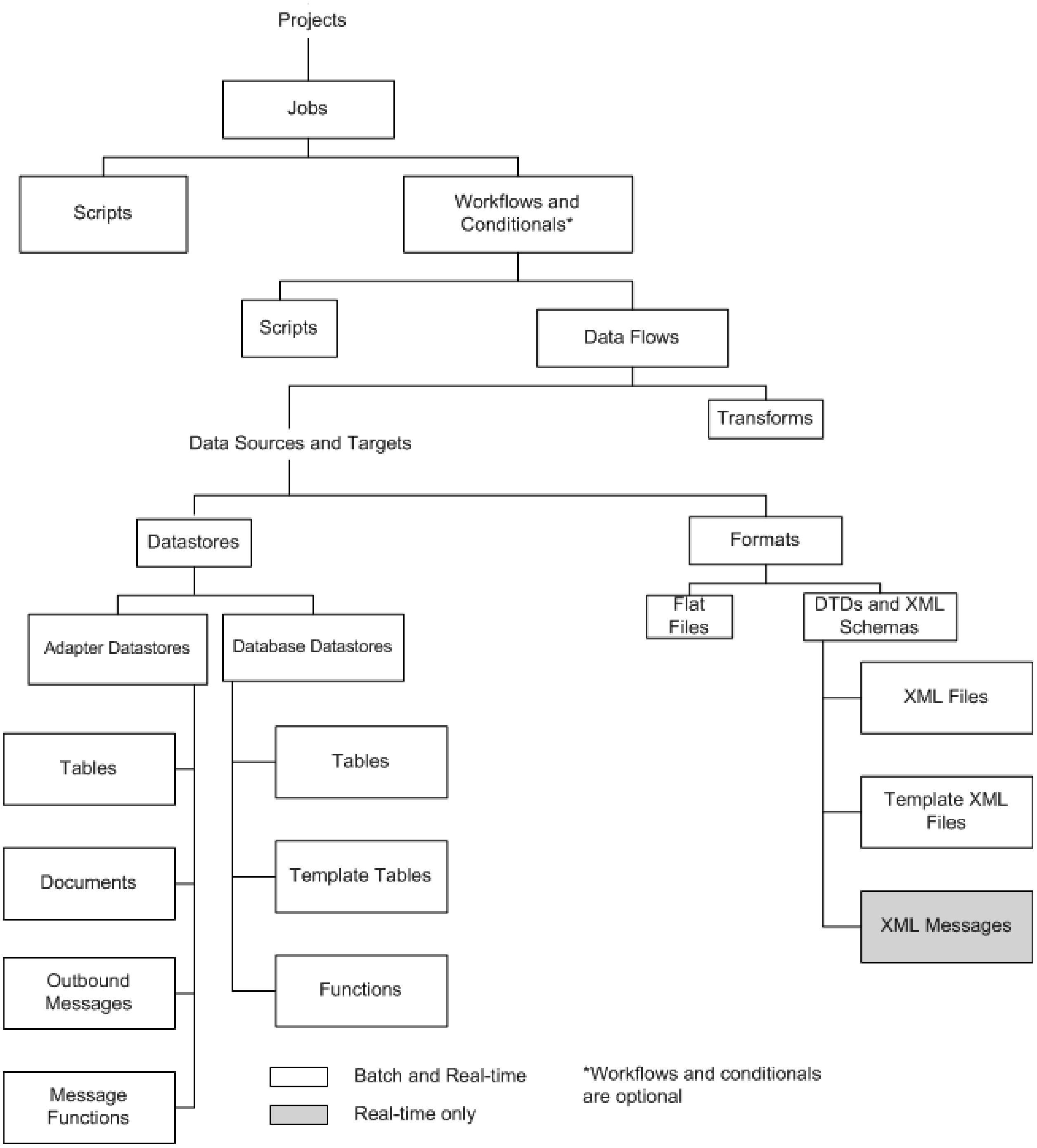
The object library contains object definitions. When you drag and drop an object from the object library, you are really creating a new reference (or call) to the existing object definition.

2.3.1.2 Single-use objects

Some objects are defined only within the context of a single job or data flow, for example scripts and specific transform definitions.

2.3.1.3 Object hierarchy

Object relationships are hierarchical. The following figure shows the relationships between major object types:



2.3.2 Menu bar

This section contains a brief description of the Designer's menus.

2.3.2.1 Project menu

The project menu contains standard Windows as well as software-specific options.

Option	Description
New	Define a new project, batch job, real-time job, work flow, data flow, transform, datastore, file format, DTD, XML Schema, or custom function.
Open	Open an existing project.
Close	Close the currently open project.
Delete	Delete the selected object.
Save	Save the object open in the workspace.
Save All	Save all changes to objects in the current Designer session.
Print	Print the active workspace.
Print Setup	Set up default printer information.
Exit	Exit Designer.

2.3.2.2 Edit menu

The Edit menu provides standard Windows commands with a few restrictions.

Option	Description
Undo	Undo the last operation.
Cut	Cut the selected objects or text and place it on the clipboard.
Copy	Copy the selected objects or text to the clipboard.
Paste	Paste the contents of the clipboard into the active workspace or text box.
Delete	Delete the selected objects.
Recover Last Deleted	Recover deleted objects to the workspace from which they were deleted. Only the most recently deleted objects are recovered.
Select All	Select all objects in the active workspace.
Clear All	Clear all objects in the active workspace (no undo).

2.3.2.3 View menu

A check mark indicates that the tool is active.

Option	Description
Toolbar	Display or remove the toolbar in the Designer window.
Status Bar	Display or remove the status bar in the Designer window.
Palette	Display or remove the floating tool palette.
Enabled Descriptions	View descriptions for objects with enabled descriptions.
Refresh	Redraw the display. Use this command to ensure the content of the workspace represents the most up-to-date information from the repository.

2.3.2.4 Tools menu

An icon with a different color background indicates that the tool is active.

Option	Description
Object Library	Open or close the object library window.
Project Area	Display or remove the project area from the Designer window.
Variables	Open or close the Variables and Parameters window.
Output	Open or close the Output window. The Output window shows errors that occur such as during job validation or object export.
Profiler Monitor	Display the status of Profiler tasks.
Run Match Wizard	Display the Match Wizard to create a match data flow. Select a transform in a data flow to activate this menu item. The transform(s) that the Match Wizard generates will be placed downstream from the transform you selected.
Match Editor	Display the Match Editor to edit Match transform options.
Associate Editor	Display the Associate Editor to edit Associate transform options.
User-Defined Editor	Display the User-Defined Editor to edit User-Defined transform options.
Custom Functions	Display the Custom Functions window.
System Configurations	Display the System Configurations editor.
Substitution Parameter Configurations	Display the Substitution Parameter Editor to create and edit substitution paramters and configurations.
Profiler Server Login	Connect to the Profiler Server.
Export	Export individual repository objects to another repository or file. This command opens the Export editor in the workspace. You can drag objects from the object library into the editor for export. To export your whole repository, in the object library right-click and select *Repository* > *Export to file*.

Option	Description
Import From File	Import objects into the current repository from a file. The default file types are ATL, XML, DMT, and FMT. For more information on DMT and FMT files, see the *Upgrade Guide*.
Metadata Exchange	Import and export metadata to third-party systems via a file.
BusinessObjects Universes	Export (create or update) metadata in BusinessObjects Universes.
Central Repositories	Create or edit connections to a central repository for managing object versions among multiple users.
Options	Display the Options window.
Data Services Management Console	Display the Management Console.

Related Information

Designer Guide: Multi-user Environment Setup [page 802]
Administrator Guide: Export/Import, Importing from a file [page 122]
Administrator Guide: Export/Import, Exporting/importing objects [page 117]
Reference Guide: Functions and Procedures, Custom functions [page 1694]
Local object library [page 186]
Project area [page 181]
Variables and Parameters [page 398]
Using the Data Profiler [page 433]
Creating and managing multiple datastore configurations [page 239]
Connecting to the profiler server [page 434]
Metadata exchange [page 731]
Creating BusinessObjects universes [page 733]
General and environment options [page 198]

2.3.2.5 Debug menu

The only options available on this menu at all times are *Show Filters/Breakpoints* and *Filters/Breakpoints*. The *Execute* and *Start Debug* options are only active when a job is selected. All other options are available as appropriate when a job is running in the Debug mode.

Option	Description
Execute	Opens the Execution Properties window which lets you execute the selected job.
Start Debug	Opens the Debug Properties window which lets you run a job in the debug mode.

Option	Description
View Design-Time Data	Opens data panes in the transform editor, which let you view and analyze the input and output for a data set in real time as you design a transform.
View Automatically	Let you view input and output data automatically after you modify a transform.
Filter Input Dataset	To filter the number of data rows displayed in the Design-Time Data Viewer panes.
Options	Opens a window in which you can configure the number of data rows displayed and the time allowed for updates before a time out.
Show Filters/Breakpoints	Shows and hides filters and breakpoints in workspace diagrams.
Filters/Breakpoints	Opens a window you can use to manage filters and breakpoints.

Related Information

Using the interactive debugger [page 701]
Using the Design-Time Data Viewer [page 699]
Filters and Breakpoints window [page 711]

2.3.2.6 Validation menu

The Designer displays options on this menu as appropriate when an object is open in the workspace.

Option	Description
Validate	Validate the objects in the current workspace view or all objects in the job before executing the application.
Show ATL	View a read-only version of the language associated with the job.
Display Optimized SQL	Display the SQL that Data Services generated for a selected data flow.

Related Information

Performance Optimization Guide: Maximizing Push-Down Operations, To view SQL [page 2121]

2.3.2.7 Window menu

The Window menu provides standard Windows options.

Option	Description
Back	Move back in the list of active workspace windows.
Forward	Move forward in the list of active workspace windows.
Cascade	Display window panels overlapping with titles showing.
Tile Horizontally	Display window panels one above the other.
Tile Vertically	Display window panels side by side.
Close All Windows	Close all open windows.

A list of objects open in the workspace also appears on the Windows menu. The name of the currently-selected object is indicated by a check mark. Navigate to another open object by selecting its name in the list.

2.3.2.8 Help menu

The Help menu provides standard help options.

Option	Description
Release Notes	Displays the *Release Notes* for this release.
What's New	Displays a summary of new features for this release.
Technical Manuals	Displays the *Technical Manuals* CHM file, a compilation of many of the Data Services technical documents. You can also access the same documentation from the <LINKDIR>\Doc \Books directory.
Tutorial	Displays the *Data Services Tutorial*, a step-by-step introduction to using SAP Data Services.
Data Services Community	Get online and timely information about SAP Data Services, including tips and tricks, additional downloads, samples, and much more. All content is to and from the community, so feel free to join in and contact us if you have a submission.
Forums on SCN (SAP Community Network)	Search the SAP forums on the SAP Community Network to learn from other SAP Data Services users and start posting questions or share your knowledge with the community.
Blueprints	Blueprints for you to download and modify to fit your needs. Each blueprint contains the necessary SAP Data Services project, jobs, data flows, file formats, sample data, template tables, and custom functions to run the data flows in your environment with only a few modifications.
Show Start Page	Displays the home page of the Data Services Designer.

Option	Description
About Data Services	Display information about the software including versions of the Designer, Job Server and engine, and copyright information.

2.3.3 Toolbar

In addition to many of the standard Windows tools, the software provides application-specific tools, including:

Icon	Tool	Description
	Close all windows	Closes all open windows in the workspace.
	Local Object Library	Opens and closes the local object library window.
	Central Object Library	Opens and closes the central object library window.
	Variables	Opens and closes the variables and parameters creation window.
	Project Area	Opens and closes the project area.
	Output	Opens and closes the output window.
	View Enabled Descriptions	Enables the system level setting for viewing object descriptions in the workspace.
	Validate Current View	Validates the object definition open in the workspace. Other objects included in the definition are also validated.
	Validate All Objects in View	Validates the object definition open in the workspace. Objects included in the definition are also validated.
✓	Audit Objects in Data Flow	Opens the Audit window to define audit labels and rules for the data flow.
	View Where Used	Opens the Output window, which lists parent objects (such as jobs) of the object currently open in the workspace (such as a data flow). Use this command to find other jobs that use the same data flow, before you decide to make design changes. To see if an object in a data flow is reused elsewhere, right-click one and select *View Where Used*.
←	Go Back	Move back in the list of active workspace windows.

Icon	Tool	Description
⇨	Go Forward	Move forward in the list of active workspace windows.
	Management Console	Opens and closes the Management Console window.
?	Contents	Opens the *Technical Manuals* PDF for information about using the software.

Use the tools to the right of the About tool with the interactive debugger.

Related Information

Debug menu options and tool bar [page 711]

2.3.4 Project area

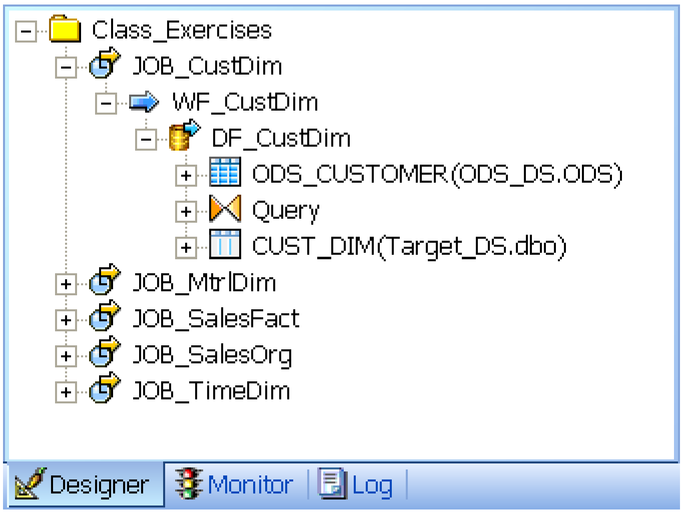
The project area provides a hierarchical view of the objects used in each project. Tabs on the bottom of the project area support different tasks. Tabs include:

Designer	Create, view and manage projects. Provides a hierarchical view of all objects used in each project.
Monitor	View the status of currently executing jobs. Selecting a specific job execution displays its status, including which steps are complete and which steps are executing. These tasks can also be done using the Administrator.
Log	View the history of complete jobs. Logs can also be viewed with the Administrator.

To control project area location, right-click its gray border and select/deselect *Docking*, or select *Hide* from the menu.

- When you select *Docking*, you can click and drag the project area to dock at and undock from any edge within the Designer window. When you drag the project area away from a Designer window edge, it stays undocked. To quickly switch between your last docked and undocked locations, just double-click the gray border.
When you deselect Allow Docking, you can click and drag the project area to any location on your screen and it will not dock inside the Designer window.
- When you select *Hide*, the project area disappears from the Designer window. To unhide the project area, click its toolbar icon.

Here's an example of the Project window's *Designer* tab, which shows the project hierarchy:



```
Class_Exercises
  JOB_CustDim
    WF_CustDim
      DF_CustDim
        ODS_CUSTOMER(ODS_DS.ODS)
        Query
        CUST_DIM(Target_DS.dbo)
  JOB_MtrlDim
  JOB_SalesFact
  JOB_SalesOrg
  JOB_TimeDim
```

As you drill down into objects in the Designer workspace, the window highlights your location within the project hierarchy.

2.3.5 Tool palette

The tool palette is a separate window that appears by default on the right edge of the Designer workspace. You can move the tool palette anywhere on your screen or dock it on any edge of the Designer window.

The icons in the tool palette allow you to create new objects in the workspace. The icons are disabled when they are not allowed to be added to the diagram open in the workspace.

To show the name of each icon, hold the cursor over the icon until the tool tip for the icon appears, as shown.

When you create an object from the tool palette, you are creating a new definition of an object. If a new object is reusable, it will be automatically available in the object library after you create it.

For example, if you select the data flow icon from the tool palette and define a new data flow, later you can drag that existing data flow from the object library, adding a call to the existing definition.

The tool palette contains the following icons:

Icon	Tool	Description (class)	Available
	Pointer	Returns the tool pointer to a selection pointer for selecting and moving objects in a diagram.	Everywhere
	Work flow	Creates a new work flow. (reusable)	Jobs and work flows
	Data flow	Creates a new data flow. (reusable)	Jobs and work flows
	ABAP data flow	Used only with the SAP application.	

Icon	Tool	Description (class)	Available
⋈	Query transform	Creates a template for a query. Use it to define column mappings and row selections. (single-use)	Data flows
▦	Template table	Creates a table for a target. (single-use)	Data flows
⇄	Template XML	Creates an XML template. (single-use)	Data flows
⇆	Data transport	Used only with the SAP application.	
📜	Script	Creates a new script object. (single-use)	Jobs and work flows
≺	Conditional	Creates a new conditional object. (single-use)	Jobs and work flows
○	Try	Creates a new try object. (single-use)	Jobs and work flows
◉	Catch	Creates a new catch object. (single-use)	Jobs and work flows
🗒	Annotation	Creates an annotation. (single-use)	Jobs, work flows, and data flows

2.3.6 Designer keyboard accessibility

The following keys are available for navigation in the Designer. All dialogs and views support these keys.

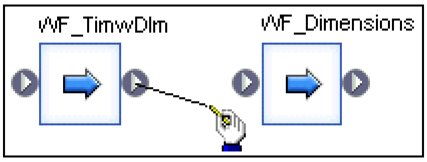
To	Press
Enter edit mode.	*F2*
Close a menu or dialog box or cancel an operation in progress.	*ESC*
Close the current window.	*CTRL+F4*
Cycle through windows one window at a time.	*CTRL+TAB*
Display a system menu for the application window.	*ALT+SPACEBAR*
Move to the next page of a property sheet.	*CTRL+PAGE DOWN*
Move to the previous page of a property sheet.	*CTRL+PAGE UP*
Move to the next control on a view or dialog.	*TAB*
Move to the previous control on a view or dialog.	*SHIFT+TAB*
Press a button when focused.	*ENTER* or *SPACE*
Enable the context menu (right-click mouse operations).	*SHIFT+F10* or *Menu Key*

To	Press
Expand or collapse a tree (+).	*Right Arrow* or *Left Arrow*
Move up and down a tree.	*Up Arrow* or *Down Arrow*
Show focus.	*ALT*
Hot Key operations.	*ALT+<LETTER>*

2.3.7 Workspace

When you open or select a job or any flow within a job hierarchy, the workspace becomes "active" with your selection. The workspace provides a place to manipulate system objects and graphically assemble data movement processes.

These processes are represented by icons that you drag and drop into a workspace to create a workspace diagram. This diagram is a visual representation of an entire data movement application or some part of a data movement application.



2.3.7.1 Moving objects in the workspace area

Use standard mouse commands to move objects in the workspace.

To move an object to a different place in the workspace area:

1. Click to select the object.
2. Drag the object to where you want to place it in the workspace.

2.3.7.2 Connecting objects

You specify the flow of data through jobs and work flows by connecting objects in the workspace from left to right in the order you want the data to be moved.

To connect objects:

1. Place the objects you want to connect in the workspace.
2. Click and drag from the triangle on the right edge of an object to the triangle on the left edge of the next object in the flow.

2.3.7.3 Disconnecting objects

To disconnect objects:

1. Click the connecting line.
2. Press the *Delete* key.

2.3.7.4 Describing objects

You can use descriptions to add comments about objects. You can use annotations to explain a job, work flow, or data flow. You can view object descriptions and annotations in the workspace. Together, descriptions and annotations allow you to document an SAP Data Services application. For example, you can describe the incremental behavior of individual jobs with numerous annotations and label each object with a basic description.

This job loads current categories and expenses and produces tables for analysis.

Related Information

Creating descriptions [page 192]
Creating annotations [page 194]

2.3.7.5 Scaling the workspace

You can control the scale of the workspace. By scaling the workspace, you can change the focus of a job, work flow, or data flow. For example, you might want to increase the scale to examine a particular part of a work flow, or you might want to reduce the scale so that you can examine the entire work flow without scrolling.

To change the scale of the workspace

1. In the drop-down list on the tool bar, select a predefined scale or enter a custom value (for example, 100%).
2. Alternatively, right-click in the workspace and select a desired scale.

> i Note
>
> You can also select *Scale to Fit* and *Scale to Whole*:

- Select *Scale to Fit* and the Designer calculates the scale that fits the entire project in the current view area.
- Select *Scale to Whole* to show the entire workspace area in the current view area.

2.3.7.6 Arranging workspace windows

The Window menu allows you to arrange multiple open workspace windows in the following ways: cascade, tile horizontally, or tile vertically.

2.3.7.7 Closing workspace windows

When you drill into an object in the project area or workspace, a view of the object's definition opens in the workspace area. The view is marked by a tab at the bottom of the workspace area, and as you open more objects in the workspace, more tabs appear. (You can show/hide these tabs from the *Tools* > *Options* menu. Go to *Designer* > *General* options and select/deselect *Show tabs in workspace*.)

> **Note**
>
> These views use system resources. If you have a large number of open views, you might notice a decline in performance.

Close the views individually by clicking the close box in the top right corner of the workspace. Close all open views by selecting *Window* > *Close All Windows* or clicking the *Close All Windows* icon on the toolbar.

Related Information

General and environment options [page 198]

2.3.8 Local object library

The local object library provides access to reusable objects. These objects include built-in system objects, such as transforms, and the objects you build and save, such as datastores, jobs, data flows, and work flows.

The local object library is a window into your local repository and eliminates the need to access the repository directly. Updates to the repository occur through normal software operation. Saving the objects you create adds them to the repository. Access saved objects through the local object library.

To control object library location, right-click its gray border and select/deselect *Docking*, or select *Hide* from the menu.

- When you select *Docking*, you can click and drag the object library to dock at and undock from any edge within the Designer window. When you drag the object library away from a Designer window edge, it stays undocked. To quickly switch between your last docked and undocked locations, just double-click the gray border.
  When you deselect *Allow Docking*, you can click and drag the object library to any location on your screen and it will not dock inside the Designer window.
- When you select *Hide*, the object library disappears from the Designer window. To unhide the object library, click its toolbar icon.

Related Information

Designer Guide: Multi-user Development, Central versus local repository [page 798]

2.3.8.1 To open the object library

Choose ▶ *Tools* ▶ *Object Library* ◀, or click the object library icon in the icon bar.

The object library gives you access to the object types listed in the following table. The table shows the tab on which the object type appears in the object library and describes the context in which you can use each type of object.

Tab	Description
Projects	Projects are sets of jobs available at a given time.
Jobs	Jobs are executable work flows. There are two job types: batch jobs and real-time jobs.
Work Flows	Work flows order data flows and the operations that support data flows, defining the interdependencies between them.
Data Flows	Data flows describe how to process a task.
Transforms	Transforms operate on data, producing output data sets from the sources you specify. The object library lists both built-in and custom transforms.
Datastores	Datastores represent connections to databases and applications used in your project. Under each datastore is a list of the tables, documents, and functions imported into the software.
Formats	Formats describe the structure of a flat file, XML file, or XML message.
Custom Functions	Custom Functions are functions written in the software's Scripting Language. You can use them in your jobs.

2.3.8.2 To display the name of each tab as well as its icon

1. Make the object library window wider until the names appear.

   or
2. Hold the cursor over the tab until the tool tip for the tab appears.

2.3.8.3 To sort columns in the object library

Click the column heading.

For example, you can sort data flows by clicking the Data Flow column heading once. Names are listed in ascending order. To list names in descending order, click the Data Flow column heading again.

2.3.9 Object editors

To work with the options for an object, in the workspace click the name of the object to open its editor. The editor displays the input and output schemas for the object and a panel below them listing options set for the object. If there are many options, they are grouped in tabs in the editor.

A schema is a data structure that can contain columns, other nested schemas, and functions (the contents are called schema elements). A table is a schema containing only columns.

In an editor, you can:

- Undo or redo previous actions performed in the window (right-click and choose *Undo* or *Redo*)
- Find a string in the editor (right-click and choose *Find*)
- Drag-and-drop column names from the input schema into relevant option boxes
- Use colors to identify strings and comments in text boxes where you can edit expressions (keywords appear blue; strings are enclosed in quotes and appear pink; comments begin with a pound sign and appear green)

> Note
>
> You cannot add comments to a mapping clause in a Query transform. For example, the following syntax is not supported on the Mapping tab:

```
table.column # comment
```

The job will not run and you cannot successfully export it. Use the object description or workspace annotation feature instead.

Related Information

Query Editor [page 311]

[page 314]

2.3.10 Working with objects

This section discusses common tasks you complete when working with objects in the Designer. With these tasks, you use various parts of the Designer—the toolbar, tool palette, workspace, and local object library.

2.3.10.1 Creating new reusable objects

You can create reusable objects from the object library or by using the tool palette. After you create an object, you can work with the object, editing its definition and adding calls to other objects.

2.3.10.1.1 To create a reusable object (in the object library)

1. Open the object library by choosing *Tools* ▸ *Object Library*.
2. Click the tab corresponding to the object type.
3. Right-click anywhere except on existing objects and choose *New*.
4. Right-click the new object and select *Properties*. Enter options such as name and description to define the object.

2.3.10.1.2 To create a reusable object (using the tool palette)

1. In the tool palette, left-click the icon for the object you want to create.
2. Move the cursor to the workspace and left-click again.

   The object icon appears in the workspace where you have clicked.

2.3.10.1.3 To open an object's definition

You can open an object's definition in one of two ways:

1. From the workspace, click the object name. The software opens a blank workspace in which you define the object.
2. From the project area, click the object.

You define an object using other objects. For example, if you click the name of a batch data flow, a new workspace opens for you to assemble sources, targets, and transforms that make up the actual flow.

2.3.10.1.4 To add an existing object (create a new call to an existing object)

1. Open the object library by choosing *Tools* ▸ *Object Library*.
2. Click the tab corresponding to any object type.
3. Select an object.
4. Drag the object to the workspace.

Note

Objects dragged into the workspace must obey the hierarchy logic. For example, you can drag a data flow into a job, but you cannot drag a work flow into a data flow.

Related Information

Object hierarchy [page 173]

2.3.10.2 Changing object names

You can change the name of an object from the workspace or the object library. You can also create a copy of an existing object.

Note

You cannot change the names of built-in objects.

1. To change the name of an object in the workspace
   a) Click to select the object in the workspace.
   b) Right-click and choose *Rename*.
   c) Edit the text in the name text box.
   d) Click outside the text box or press Enter to save the new name.
2. To change the name of an object in the object library
   a) Select the object in the object library.
   b) Right-click and choose *Properties*.
   c) Edit the text in the first text box.
   d) Click *OK*.
3. To copy an object
   a) Select the object in the object library.
   b) Right-click and choose *Replicate*.
   c) The software makes a copy of the top-level object (but not of objects that it calls) and gives it a new name, which you can edit.

2.3.10.3 Viewing and changing object properties

You can view (and, in some cases, change) an object's properties through its property page.

2.3.10.3.1 To view, change, and add object properties

1. Select the object in the object library.
2. Right-click and choose *Properties*. The *General* tab of the Properties window opens.
3. Complete the property sheets. The property sheets vary by object type, but General, Attributes and Class Attributes are the most common and are described in the following sections.
4. When finished, click *OK* to save changes you made to the object properties and to close the window.

   Alternatively, click Apply to save changes without closing the window.

2.3.10.3.2 General tab

The *General* tab contains two main object properties: name and description.

From the General tab, you can change the object name as well as enter or edit the object description. You can add object descriptions to single-use objects as well as to reusable objects. Note that you can toggle object descriptions on and off by right-clicking any object in the workspace and selecting/clearing *View Enabled Descriptions*.

Depending on the object, other properties may appear on the *General* tab. Examples include:

- *Execute only once*
- *Recover as a unit*
- *Degree of parallelism*
- *Use database links*
- *Cache type*

Related Information

Performance Optimization Guide: Using Caches [page 2129]

Linked datastores [page 231]

Performance Optimization Guide: Using Parallel Execution [page 2137]

Recovery Mechanisms [page 720]

Creating and defining data flows [page 284]

2.3.10.3.3 Attributes tab

The *Attributes* tab allows you to assign values to the attributes of the current object.

To assign a value to an attribute, select the attribute and enter the value in the *Value* box at the bottom of the window.

Some attribute values are set by the software and cannot be edited. When you select an attribute with a system-defined value, the *Value* field is unavailable.

2.3.10.3.4 Class Attributes tab

The *Class Attributes* tab shows the attributes available for the type of object selected. For example, all data flow objects have the same class attributes.

To create a new attribute for a class of objects, right-click in the attribute list and select *Add*. The new attribute is now available for all of the objects of this class.

To delete an attribute, select it then right-click and choose *Delete*. You cannot delete the class attributes predefined by Data Services.

2.3.10.4 Creating descriptions

Use descriptions to document objects. You can see descriptions on workspace diagrams. Therefore, descriptions are a convenient way to add comments to workspace objects.

A description is associated with a particular object. When you import or export that repository object (for example, when migrating between development, test, and production environments), you also import or export its description.

The Designer determines when to show object descriptions based on a system-level setting and an object-level setting. Both settings must be activated to view the description for a particular object.

The system-level setting is unique to your setup. The system-level setting is disabled by default. To activate that system-level setting, select *ViewEnabled Descriptions*, or click the *View Enabled Descriptions* button on the toolbar.

The object-level setting is saved with the object in the repository. The object-level setting is also disabled by default unless you add or edit a description from the workspace. To activate the object-level setting, right-click the object and select *Enable object description*.

An ellipses after the text in a description indicates that there is more text. To see all the text, resize the description by clicking and dragging it. When you move an object, its description moves as well. To see which object is associated with which selected description, view the object's name in the status bar.

2.3.10.4.1 To add a description to an object

1. In the project area or object library, right-click an object and select *Properties*.
2. Enter your comments in the *Description* text box.
3. Click *OK*.

   The description for the object displays in the object library.

2.3.10.4.2 To display a description in the workspace

1. In the project area, select an existing object (such as a job) that contains an object to which you have added a description (such as a work flow).
2. From the *View* menu, select *Enabled Descriptions*.

   Alternately, you can select the View Enabled Descriptions button on the toolbar.
3. Right-click the work flow and select *Enable Object Description*.

   The description displays in the workspace under the object.

2.3.10.4.3 To add a description to an object from the workspace

1. From the *View* menu, select *Enabled Descriptions*.
2. In the workspace, right-click an object and select *Properties*.
3. In the Properties window, enter text in the *Description* box.
4. Click *OK*.

   The description displays automatically in the workspace (and the object's Enable Object Description option is selected).

2.3.10.4.4 To hide an object's description

1. In the workspace diagram, right-click an object.

   Alternately, you can select multiple objects by:
   - Pressing and holding the Control key while selecting objects in the workspace diagram, then right-clicking one of the selected objects.
   - Dragging a selection box around all the objects you want to select, then right-clicking one of the selected objects.
2. In the pop-up menu, deselect *Enable Object Description*.

   The description for the object selected is hidden, even if the View Enabled Descriptions option is checked, because the object-level switch overrides the system-level switch.

2.3.10.4.5 To edit object descriptions

1. In the workspace, double-click an object description.
2. Enter, cut, copy, or paste text into the description.
3. In the *Project* menu, select *Save*.

Alternately, you can right-click any object and select Properties to open the object's Properties window and add or edit its description.

> **Note**
>
> If you attempt to edit the description of a reusable object, the software alerts you that the description will be updated for every occurrence of the object, across all jobs. You can select the *Don't show this warning next time* check box to avoid this alert. However, after deactivating the alert, you can only reactivate the alert by calling Technical Support.

2.3.10.5 Creating annotations

Annotations describe a flow, part of a flow, or a diagram in a workspace. An annotation is associated with the job, work flow, or data flow where it appears. When you import or export that job, work flow, or data flow, you import or export associated annotations.

2.3.10.5.1 To annotate a workspace diagram

1. Open the workspace diagram you want to annotate.

   You can use annotations to describe any workspace such as a job, work flow, data flow, catch, conditional, or while loop.
2. In the tool palette, click the annotation icon.
3. Click a location in the workspace to place the annotation.

   An annotation appears on the diagram.

   You can add, edit, and delete text directly on the annotation. In addition, you can resize and move the annotation by clicking and dragging. You can add any number of annotations to a diagram.

2.3.10.5.2 To delete an annotation

1. Right-click an annotation.
2. Select *Delete*.

   Alternately, you can select an annotation and press the Delete key.

2.3.10.6 Copying objects

Objects can be cut or copied and then pasted on the workspace where valid. Multiple objects can be copied and pasted either within the same or other data flows, work flows, or jobs. Additionally, calls to data flows and works flows can be cut or copied and then pasted to valid objects in the workspace.

References to global variables, local variables, parameters, and substitution parameters are copied; however, you must be define each within its new context.

> Note
>
> The paste operation duplicates the selected objects in a flow, but still calls the original objects. In other words, the paste operation uses the original object in another location. The replicate operation creates a new object in the object library.

To cut or copy and then paste objects:

1. In the workspace, select the objects you want to cut or copy.

   You can select multiple objects using Ctrl-click, Shift-click, or Ctrl+A.
2. Right-click and then select either *Cut* or *Copy*.
3. Click within the same flow or select a different flow. Right-click and select *Paste*.

   Where necessary to avoid a naming conflict, a new name is automatically generated.

   > Note
   >
   > The objects are pasted in the selected location if you right-click and select *Paste*.
   >
   > The objects are pasted in the upper left-hand corner of the workspace if you paste using any of the following methods:
   >
   > ○ click the *Paste* icon.
   >
   > ○ click ▶ *Edit* › *Paste* ◀.
   >
   > ○ use the Ctrl+V keyboard short-cut.
   >
   > If you use a method that pastes the objects to the upper left-hand corner, subsequent pasted objects are layered on top of each other.

2.3.10.7 Saving and deleting objects

Saving an object in the software means storing the language that describes the object to the repository. You can save reusable objects; single-use objects are saved only as part of the definition of the reusable object that calls them.

You can choose to save changes to the reusable object currently open in the workspace. When you save the object, the object properties, the definitions of any single-use objects it calls, and any calls to other reusable objects are recorded in the repository. The content of the included reusable objects is not saved; only the call is saved.

The software stores the description even if the object is not complete or contains an error (does not validate).

2.3.10.7.1 To save changes to a single reusable object

1. Open the project in which your object is included.

2. Choose ▸ *Project* ▸ *Save* ◂.
   This command saves all objects open in the workspace.

Repeat these steps for other individual objects you want to save.

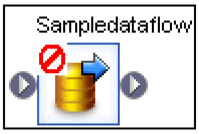
2.3.10.7.2 To save all changed objects in the repository

1. Choose ▸ *Project* ▸ *Save All* ◂.
   The software lists the reusable objects that were changed since the last save operation.
2. (optional) Deselect any listed object to avoid saving it.
3. Click *OK*.

> **Note**
>
> The software also prompts you to save all objects that have changes when you execute a job and when you exit the Designer. Saving a reusable object saves any single-use object included in it.

2.3.10.7.3 To delete an object definition from the repository

1. In the object library, select the object.
2. Right-click and choose *Delete*.
   - If you attempt to delete an object that is being used, the software provides a warning message and the option of using the *View Where Used* feature.
   - If you select *Yes*, the software marks all calls to the object with a red “deleted” icon to indicate that the calls are invalid. You must remove or replace these calls to produce an executable job.



> **Note**
>
> Built-in objects such as transforms cannot be deleted from the object library.

Related Information

Using View Where Used [page 686]

2.3.10.7.4 To delete an object call

1. Open the object that contains the call you want to delete.
2. Right-click the object call and choose *Delete*.

If you delete a reusable object from the workspace or from the project area, only the object call is deleted. The object definition remains in the object library.

2.3.10.8 Searching for objects

From within the object library, you can search for objects defined in the repository or objects available through a datastore.

2.3.10.8.1 To search for an object

1. Right-click in the object library and choose *Search*.

   The software displays the Search window.
2. Enter the appropriate values for the search.

   Options available in the Search window are described in detail following this procedure.
3. Click *Search*.

   The objects matching your entries are listed in the window. From the search results window you can use the context menu to:

   - *Open* an item
   - *View* the attributes (*Properties*)
   - *Import* external tables as repository metadata

   You can also drag objects from the search results window and drop them in the desired location.

The Search window provides you with the following options:

Option	Description
Look in	Where to search. Choose from the repository or a specific datastore. When you designate a datastore, you can also choose to search the imported data (*Internal Data*) or the entire datastore (*External Data*).
Object type	The type of object to find. When searching the repository, choose from Tables, Files, Data flows, Work flows, Jobs, Hierarchies, IDOCs, and Domains. When searching a datastore or application, choose from object types available through that datastore.

Option	Description
Name	The object name to find. If you are searching in the repository, the name is not case sensitive. If you are searching in a datastore and the name is case sensitive in that datastore, enter the name as it appears in the database or application and use double quotation marks (") around the name to preserve the case. You can designate whether the information to be located *Contains* the specified name or *Equals* the specified name using the drop-down box next to the *Name* field.
Description	The object description to find. Objects imported into the repository have a description from their source. By default, objects you create in the Designer have no description unless you add a one. The search returns objects whose description attribute contains the value entered.

The Search window also includes an *Advanced* button where, you can choose to search for objects based on their attribute values. You can search by attribute values only when searching in the repository.

The *Advanced* button provides the following options:

Option	Description
Attribute	The object attribute in which to search.
Value	The attribute value to find.
Match	The type of search performed. Select *Contains* to search for any attribute that contains the value specified. Select *Equals* to search for any attribute that contains only the value specified.

2.3.11 General and environment options

To open the *Options* window, select ▶ *Tools* ▸ *Options* ◢. The window displays option groups for Designer, Data, and Job Server options.

Expand the options by clicking the plus icon. As you select each option group or option, a description appears on the right.

2.3.11.1 Designer — Environment

Table 1: Default Administrator for Metadata Reporting

Option	Description
Administrator	Select the Administrator that the metadata reporting tool uses. An Administrator is defined by host name and port.

Table 2: Default Job Server

Option	Description
Current	Displays the current value of the default Job Server.
New	Allows you to specify a new value for the default Job Server from a drop-down list of Job Servers associated with this repository. Changes are effective immediately.

If a repository is associated with several Job Servers, one Job Server must be defined as the default Job Server to use at login.

> Note
>
> Job-specific options and path names specified in Designer refer to the current default Job Server. If you change the default Job Server, modify these options and path names.

Table 3: Designer Communication Ports

Option	Description
Allow Designer to set the port for Job Server communication	If checked, Designer automatically sets an available port to receive messages from the current Job Server. The default is checked. Uncheck to specify a listening port or port range.
From *To*	Enter port numbers in the port text boxes. To specify a specific listening port, enter the same port number in both the *From* port and *To* port text boxes. Changes will not take effect until you restart the software. Only activated when you deselect the previous control. Allows you to specify a range of ports from which the Designer can choose a listening port. You may choose to constrain the port used for communication between Designer and Job Server when the two components are separated by a firewall.
Interactive Debugger	Allows you to set a communication port for the Designer to communicate with a Job Server while running in Debug mode.
Server group for local repository	If the local repository that you logged in to when you opened the Designer is associated with a server group, the name of the server group appears.

Related Information

Changing the interactive debugger port [page 704]

2.3.11.2 Designer — General

Option	Description
View data sampling size (rows)	Controls the sample size used to display the data in sources and targets in open data flows in the workspace. View data by clicking the magnifying glass icon on source and target objects.
Number of characters in workspace icon name	Controls the length of the object names displayed in the workspace. Object names are allowed to exceed this number, but the Designer only displays the number entered here. The default is 17 characters.
Maximum schema tree elements to auto expand	The number of elements displayed in the schema tree. Element names are not allowed to exceed this number. Enter a number for the *Input schema* and the *Output schema*. The default is 100.
Default parameters to variables of the same name	When you declare a variable at the work-flow level, the software automatically passes the value as a parameter with the same name to a data flow called by a work flow.
Automatically import domains	Select this check box to automatically import domains when importing a table that references a domain.
Perform complete validation before job execution	If checked, the software performs a complete job validation before running a job. The default is unchecked. If you keep this default setting, you should validate your design manually before job execution.
Open monitor on job execution	Affects the behavior of the Designer when you execute a job. With this option enabled, the Designer switches the workspace to the monitor view during job execution; otherwise, the workspace remains as is. The default is on.
Automatically calculate column mappings	Calculates information about target tables and columns and the sources used to populate them. The software uses this information for metadata reports such as impact and lineage, auto documentation, or custom reports. Column mapping information is stored in the AL_COLMAP table (ALVW_MAPPING view) after you save a data flow or import objects to or export objects from a repository. If the option is selected, be sure to validate your entire job before saving it because column mapping calculation is sensitive to errors and will skip data flows that have validation problems.
Show dialog when job is completed	Allows you to choose if you want to see an alert or just read the trace messages.

Option	Description
Show tabs in workspace	Allows you to decide if you want to use the tabs at the bottom of the workspace to navigate.
Single window in workspace	Allows you to view only one window in the workspace area (for example, a job or transform). If you open another window, the previous window closes and the new window opens.
Show Start Page at startup	Allows you to view the Designer start page when you open the Designer.
Enable Object Description when instantiate	Enables element descriptions for elements that you place in your flow (for example, a reader or transform that you put into a job, workflow, or data flow).
Exclude non-executable elements from export to XML Document	Excludes elements not processed during job execution from exported XML documents. For example, Designer workspace display coordinates would not be exported.

Related Information

Using View Data [page 688]

Management Console Guide: Refresh Usage Data tab [page 1974]

2.3.11.3 Designer — Graphics

Choose and preview stylistic elements to customize your workspaces. Using these options, you can easily distinguish your job/work flow design workspace from your data flow design workspace.

Option	Description
Workspace flow type	Switch between the two workspace flow types (Job/Work Flow and Data Flow) to view default settings. Modify settings for each type using the remaining options.
Line Type	Choose a style for object connector lines.
Line Thickness	Set the connector line thickness.
Background style	Choose a plain or tiled background pattern for the selected flow type.
Color scheme	Set the background color to blue, gray, or white.
Use navigation watermark	Add a watermark graphic to the background of the flow type selected. Note that this option is only available with a plain background style.

2.3.11.4 Designer — Attribute values

When re-importing, Data Services preserves or clears the old value for some table and column attributes.

You can control what Data Services does on re-import for the following attributes:

Attribute	Object type	Description
Associated_Dimension	Column	The name of the dimension attached to the detail. Used to support the metadata exchanged between SAP Data Services and SAP Universe Builder.
Business_Description	Column and Table	A business-level description of a table or column.
Business_Name	Column and Table	A logical field. This attribute defines and runs jobs that extract, transform, and load physical data while the Business Name data remains intact.
Column_Usage	Column	Supports metadata exchanged between SAP Data Services and SAP Universe Builder.
Content_Type	Column	Defines the type of data in a column.
Description	Column and Table	Description of the column or table.
Estimated_Row_Count	Table	An estimate of the table size used in calculating the order in which tables are read to perform join operations.
ObjectLabel	Column and Table	A label used to describe an object.
Table_Usage	Table	A label field used to mark a table as fact or dimension, for example.

2.3.11.5 Designer — Central Repository Connections

Option	Description
Central Repository Connections	Displays the central repository connections and the active central repository. To activate a central repository, right-click one of the central repository connections listed and select *Activate*.
Reactivate automatically	Select if you want the active central repository to be reactivated whenever you log in to the software using the current local repository.

2.3.11.6 Designer — Language

These options provide you with locales, other than English (the default locale), for viewing the Data Services user interface and any text that the user interface generaties in other languages.

You can select the locale for both the user interface and the displayed data.

Option	Description
Product locale	Specifies the user interface language and all product messages.
Preferred viewing locale	Specifies the locale that the user data should be presented in. For example, date formatting should be presented in the preferred viewing locale.

2.3.11.7 Designer — SSL

By default, the paths for the SSL certificate and keyfiles are automatically configured during installation. You do not need to change them unless you want to use your own certificates.

Note

If you change any SSL options other than *Use SSL protocol for profiler*, you must restart both the Designer and any Data Services servers.

Option	Description
Server certificate file	The path to the server certificate file. The server certificate must be in PEM format.
Server private key file	The path to the server private key file.
Use server private key password file	Select this option and specify the location of the password file if you want to use a private key password file.
Trusted certificates folder	Specify the location of the trusted certificates folder. Valid extensions for certificates in the trusted certificates folder include .pem, .crt, and .cer. Regardless of the file extension, all certificate file contents must be in PEM format.
Use SSL protocol for profiler	Select this option to use SSL protocol for communications between the Designer and the profiler server.

2.3.11.8 Data — General

Option	Description
Century Change Year	Indicates how the software interprets the century for two-digit years. Two-digit years greater than or equal to this value are interpreted as 19##. Two-digit years less than this value are interpreted as 20##. The default value is 15. For example, if the *Century Change Year* is set to 15:

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><table><tr><th>Two-digit year</th><th>Interpreted as</th></tr><tr><td>99</td><td>1999</td></tr><tr><td>16</td><td>1916</td></tr><tr><td>15</td><td>1915</td></tr><tr><td>14</td><td>2014</td></tr></table></td></tr>
<tr><td><i>Convert blanks to nulls for Oracle bulk loader</i></td><td>Converts blanks to NULL values when loading data using the Oracle bulk loader utility and:
● the column is not part of the primary key
● the column is nullable</td></tr>
</table>

2.3.11.9 Job Server — Environment

Option	Description
Maximum number of engine processes	Sets a limit on the number of engine processes that this Job Server can have running concurrently.

2.3.11.10 Job Server — General

Use this window to reset Job Server options or with guidance from SAP Technical customer Support.

Related Information

Changing Job Server options [page 428]

2.4 Projects and Jobs

Project and job objects represent the top two levels of organization for the application flows you create using the Designer.

2.4.1 Projects

A project is a reusable object that allows you to group jobs. A project is the highest level of organization offered by the software. Opening a project makes one group of objects easily accessible in the user interface.

You can use a project to group jobs that have schedules that depend on one another or that you want to monitor together.

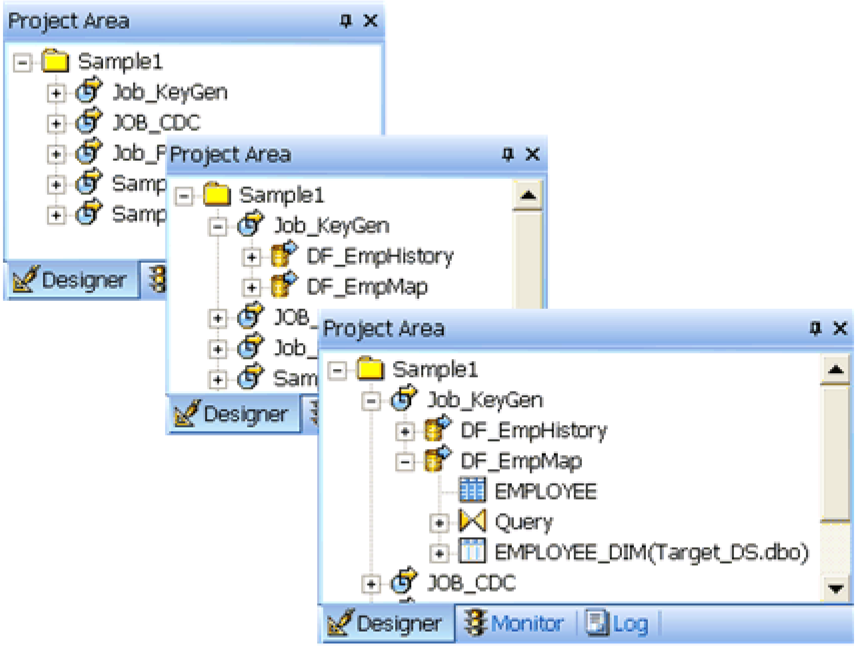
Projects have common characteristics:

- Projects are listed in the object library.
- Only one project can be open at a time.
- Projects cannot be shared among multiple users.

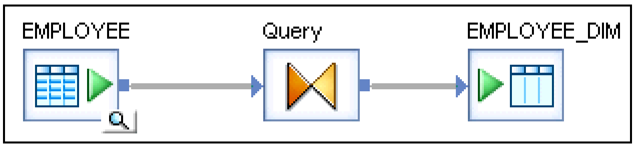
2.4.1.1 Objects that make up a project

The objects in a project appear hierarchically in the project area. If a plus sign (+) appears next to an object, expand it to view the lower-level objects contained in the object. The software shows you the contents as both names in the project area hierarchy and icons in the workspace.

In the following example, the Job_KeyGen job contains two data flows, and the DF_EmpMap data flow contains multiple objects.



Each item selected in the project area also displays in the workspace:



2.4.1.2 Creating a new project

1. Choose ▶ *Project* ▶ *New* ▶ *Project* ◀.
2. Enter the name of your new project.

   The name can include alphanumeric characters and underscores (_). It cannot contain blank spaces.
3. Click *Create*.

The new project appears in the project area. As you add jobs and other lower-level objects to the project, they also appear in the project area.

2.4.1.3 Opening existing projects

2.4.1.3.1 To open an existing project

1. Choose ▶ *Project* ▶ *Open* ◀.
2. Select the name of an existing project from the list.
3. Click *Open*.

> i Note
>
> If another project was already open, the software closes that project and opens the new one.

2.4.1.4 Saving projects

2.4.1.4.1 To save all changes to a project

1. Choose ▶ *Project* ▶ *Save All* ◀.

   The software lists the jobs, work flows, and data flows that you edited since the last save.
2. (optional) Deselect any listed object to avoid saving it.
3. Click *OK*.

   > i Note
   >
   > The software also prompts you to save all objects that have changes when you execute a job and when you exit the Designer. Saving a reusable object saves any single-use object included in it.

2.4.2 Jobs

A job is the only object you can execute. You can manually execute and test jobs in development. In production, you can schedule batch jobs and set up real-time jobs as services that execute a process when the software receives a message request.

A job is made up of steps you want executed together. Each step is represented by an object icon that you place in the workspace to create a job diagram. A job diagram is made up of two or more objects connected together. You can include any of the following objects in a job definition:

- Data flows
    - Sources
    - Targets
    - Transforms
- Work flows
    - Scripts
    - Conditionals
    - While Loops
    - Try/catch blocks

If a job becomes complex, organize its content into individual work flows, then create a single job that calls those work flows.

Real-time jobs use the same components as batch jobs. You can add work flows and data flows to both batch and real-time jobs. When you drag a work flow or data flow icon into a job, you are telling the software to validate these objects according the requirements of the job type (either batch or real-time).

There are some restrictions regarding the use of some software features with real-time jobs.

Related Information

Work Flows [page 328]

Real-time Jobs [page 372]

2.4.2.1 Creating jobs

2.4.2.1.1 To create a job in the project area

1. In the project area, select the project name.
2. Right-click and choose *New Batch Job* or *Real Time Job*.
3. Edit the name.

The name can include alphanumeric characters and underscores (_). It cannot contain blank spaces.

The software opens a new workspace for you to define the job.

2.4.2.1.2 To create a job in the object library

1. Go to the *Jobs* tab.
2. Right-click *Batch Jobs* or *Real Time Jobs* and choose *New*.
3. A new job with a default name appears.
4. Right-click and select Properties to change the object's name and add a description.

   The name can include alphanumeric characters and underscores (_). It cannot contain blank spaces.
5. To add the job to the open project, drag it into the project area.

2.4.2.2 Naming conventions for objects in jobs

We recommend that you follow consistent naming conventions to facilitate object identification across all systems in your enterprise. This allows you to more easily work with metadata across all applications such as:

- Data-modeling applications
- ETL applications
- Reporting applications
- Adapter software development kits

Examples of conventions recommended for use with jobs and other objects are shown in the following table.

Prefix	Suffix	Object	Example
DF_	n/a	Data flow	DF_Currency
EDF_	_Input	Embedded data flow	EDF_Example_Input
EDF_	_Output	Embedded data flow	EDF_Example_Output
RTJob_	n/a	Real-time job	RTJob_OrderStatus
WF_	n/a	Work flow	WF_SalesOrg
JOB_	n/a	Job	JOB_SalesOrg
n/a	_DS	Datastore	ORA_DS
DC_	n/a	Datastore configuration	DC_DB2_production
SC_	n/a	System configuration	SC_ORA_test
n/a	_Memory_DS	Memory datastore	Catalog_Memory_DS
PROC_	n/a	Stored procedure	PROC_SalesStatus

Although the Designer is a graphical user interface with icons representing objects in its windows, other interfaces might require you to identify object types by the text alone. By using a prefix or suffix, you can more easily identify your object's type.

In addition to prefixes and suffixes, you might want to provide standardized names for objects that identify a specific action across all object types. For example: DF_OrderStatus, RTJob_OrderStatus.

In addition to prefixes and suffixes, naming conventions can also include path name identifiers. For example, the stored procedure naming convention can look like either of the following:

<datastore>.<owner>.<PROC_Name>

<datastore>.<owner>.<package>.<PROC_Name>

2.5 Datastores

This section describes different types of datastores, provides details about the Attunity Connector datastore, and instructions for configuring datastores.

2.5.1 What are datastores?

Datastores represent connection configurations between the software and databases or applications. These configurations can be direct or through adapters. Datastore configurations allow the software to access metadata from a database or application and read from or write to that database or application while the software executes a job.

SAP Data Services datastores can connect to:

- Databases and mainframe file systems.
- Applications that have pre-packaged or user-written adapters.
- J.D. Edwards One World and J.D. Edwards World, Oracle Applications, PeopleSoft, SAP Applications, SAP Master Data Services, SAP NetWeaver BW, and Siebel Applications. See the appropriate supplement guide.

> Note
>
> The software reads and writes data stored in flat files through flat file formats. The software reads and writes data stored in XML documents through DTDs and XML Schemas.

The specific information that a datastore object can access depends on the connection configuration. When your database or application changes, make corresponding changes in the datastore information in the software. The software does not automatically detect the new information.

> Note
>
> Objects deleted from a datastore connection are identified in the project area and workspace by a red "deleted" icon. 🚫 This visual flag allows you to find and update data flows affected by datastore changes.

You can create multiple configurations for a datastore. This allows you to plan ahead for the different environments your datastore may be used in and limits the work involved with migrating jobs. For example, you can add a set of configurations (DEV, TEST, and PROD) to the same datastore name. These connection settings stay with the datastore during export or import. You can group any set of datastore configurations into a system configuration. When running or scheduling a job, select a system configuration, and thus, the set of datastore configurations for your current environment.

Related Information

Database datastores [page 210]
Adapter datastores [page 233]
File Formats [page 254]
Formatting XML documents [page 345]
Creating and managing multiple datastore configurations [page 239]

2.5.2 Database datastores

Database datastores can represent single or multiple connections with:

- Legacy systems using Attunity Connect
- IBM DB2, HP Neoview, Informix, Microsoft SQL Server, Oracle, SQL Anywhere, Sybase ASE, Sybase IQ, MySQL, Netezza, SAP HANA, SAP Data Federator, SQL Anywhere and Teradata databases (using native connections)
- Other databases (through ODBC)
- A repository, using a memory datastore or persistent cache datastore

You can create a connection to most of the data sources using the server name instead of the DSN (Data Source Name) or TNS (Transparent Network Substrate) name. Server name connections (also known as DSN-less and TNS-less connections) eliminate the need to configure the same DSN or TNS entries on every machine in a distributed environment.

Related Information

Administrator Guide: DSN-less and TNS-less connections [page 30]

2.5.2.1 Mainframe interface

The software provides the Attunity Connector datastore that accesses mainframe data sources through Attunity Connect. The data sources that Attunity Connect accesses are in the following list. For a complete list of sources, refer to the Attunity documentation.

- Adabas
- DB2 UDB for OS/390 and DB2 UDB for OS/400
- IMS/DB

- VSAM
- Flat files on OS/390 and flat files on OS/400

2.5.2.1.1 Prerequisites for an Attunity datastore

Attunity Connector accesses mainframe data using software that you must manually install on the mainframe server and the local client (Job Server) computer. The software connects to Attunity Connector using its ODBC interface.

It is not necessary to purchase a separate ODBC driver manager for UNIX and Windows platforms.

Servers

Install and configure the Attunity Connect product on the server (for example, an zSeries computer).

Clients

To access mainframe data using Attunity Connector, install the Attunity Connect product. The ODBC driver is required. Attunity also offers an optional tool called Attunity Studio, which you can use for configuration and administration.

Configure ODBC data sources on the client (SAP Data Services Job Server).

When you install a Job Server on UNIX, the installer will prompt you to provide an installation directory path for Attunity connector software. In addition, you do not need to install a driver manager, because the software loads ODBC drivers directly on UNIX platforms.

For more information about how to install and configure these products, refer to their documentation.

2.5.2.1.2 Configuring an Attunity datastore

To use the Attunity Connector datastore option, upgrade your repository to SAP Data Services version 6.5.1 or later.

To create an Attunity Connector datastore:

1. In the *Datastores* tab of the object library, right-click and select *New*.
2. Enter a name for the datastore.
3. In the *Datastore type* box, select *Database*.
4. In the *Database type* box, select *Attunity Connector*.
5. Type the Attunity data source name, location of the Attunity daemon (*Host location*), the Attunity daemon port number, and a unique Attunity server workspace name.

6. To change any of the default options (such as *Rows per Commit* or *Language*), click the *Advanced* button.
7. Click *OK*.

   You can now use the new datastore connection to import metadata tables into the current repository.

2.5.2.1.3 Specifying multiple data sources in one Attunity datastore

You can use the Attunity Connector datastore to access multiple Attunity data sources on the same Attunity Daemon location. If you have several types of data on the same computer, for example a DB2 database and VSAM, you might want to access both types of data using a single connection. For example, you can use a single connection to join tables (and push the join operation down to a remote server), which reduces the amount of data transmitted through your network.

To specify multiple sources in the Datastore Editor:

1. Separate data source names with semicolons in the Attunity data source box using the following format:

   ```
   AttunityDataSourceName;AttunityDataSourceName
   ```

   For example, if you have a DB2 data source named DSN4 and a VSAM data source named Navdemo, enter the following values into the Data source box:

   ```
   DSN4;Navdemo
   ```
2. If you list multiple data source names for one Attunity Connector datastore, ensure that you meet the following requirements:
   - All Attunity data sources must be accessible by the same user name and password.
   - All Attunity data sources must use the same workspace. When you setup access to the data sources in Attunity Studio, use the same workspace name for each data source.

2.5.2.1.4 Data Services naming convention for Attunity tables

Data Services' format for accessing Attunity tables is unique to Data Services. Because a single datastore can access multiple software systems that do not share the same namespace, the name of the Attunity data source must be specified when referring to a table. With an Attunity Connector, precede the table name with the data source and owner names separated by a colon. The format is as follows:

```
AttunityDataSource:OwnerName.TableName
```

When using the Designer to create your jobs with imported Attunity tables, Data Services automatically generates the correct SQL for this format. However, when you author SQL, be sure to use this format. You can author SQL in the following constructs:

- SQL function
- SQL transform
- Pushdown_sql function
- Pre-load commands in table loader
- Post-load commands in table loader

Note

For tables in Data Services, the maximum length of the owner name for most repository types is 256 (MySQL is 64 and MS SQL server is 128). In the case of Attunity tables, the maximum length of the Attunity data source name and actual owner name is 63 (the colon accounts for 1 character).

2.5.2.1.5 Limitations

All Data Services features are available when you use an Attunity Connector datastore except the following:

- Bulk loading
- Imported functions (imports metadata for tables only)
- Template tables (creating tables)
- The datetime data type supports up to 2 sub-seconds only
- Data Services cannot load timestamp data into a timestamp column in a table because Attunity truncates varchar data to 8 characters, which is not enough to correctly represent a timestamp value.
- When running a job on UNIX, the job could fail with following error:

  ```
  [D000] Cannot open file /usr1/attun/navroot/def/sys System error 13: The file
  access permissions do not allow the specified action.; (OPEN)
  ```

  This error occurs because of insufficient file permissions to some of the files in the Attunity installation directory. To avoid this error, change the file permissions for all files in the Attunity directory to 777 by executing the following command from the Attunity installation directory:

  ```
  $ chmod -R 777 *
  ```

2.5.2.2 Defining a database datastore

Define at least one database datastore for each database or mainframe file system with which you are exchanging data.

To define a datastore, get appropriate access privileges to the database or file system that the datastore describes.

For example, to allow the software to use parameterized SQL when reading or writing to DB2 databases, authorize the user (of the datastore/database) to create, execute, and drop stored procedures. If a user is not authorized to create, execute, and drop stored procedures, jobs will still run. However, the jobs will produce a warning message and will run less efficiently.

2.5.2.2.1 To define a Database datastore

1. In the *Datastores* tab of the object library, right-click and select *New*.
2. Enter the name of the new datastore in the *Datastore Name* field.

   The name can contain any alphabetical or numeric characters or underscores (_). It cannot contain spaces.
3. Select the *Datastore type*.

   When you select a Datastore Type, the software displays other options relevant to that type.
4. Select the *Database type*.
5. If you selected Oracle, the default connection type is TNS-less (the *Use TNS name* checkbox is not selected).
   - If you want to use a TNS-less connection, type in the *Hostname*, *SID*, and *Port*
   - If you want to use a TNS connection, select the check box for *Use TNS name* and type in the *TNS name*.
6. If you selected a database type that supports DSN-less connections (such as DB2, Informix, MySQL, Netezza, or SAP HANA), the default connection type is DSN-less (the *Use data source name (DSN)* checkbox is not selected).
   - If you want to use a DSN-less connection, type in the *Database server name*, *Database name* (for DB2 and MySQL) and *Port*.

     > Note
     >
     > If you selected Informix, you must specify the server instance name (not the host name) in *Database server name*. If Data Services is on a different computer than the Informix server, follow the next step.
   - If you want to use a DSN connection, select the check box for *Use data source name (DSN)* and type in the *Data source name*.

   For the most current list of supported databases for DSN-less connections, see the *Release Notes*.
7. If you selected Informix, want to use DSN-less connections, and Data Services is on a different computer than the Informix server, you must identify the Informix host as follows:

   a) Go to your Informix client installation folder.
   For example: `C:\Program Files\IBM\Informix\Client-SDK\bin`

   b) Run **`setnet32.exe`** and go to the *Server Information* tab.

   c) Enter the IBM Informix Server name, Host name, and other required information.

   d) Click *Make Default Server*, and then click *OK*.

   For DSN-less connections to an Informix database, the Designer can now obtain the Informix host name for the Informix server name you provided.
8. If you selected Data Federator, you must also specify the catalog name and the schema name in the URL. If you do not, you may see all of the tables from each catalog.

   a) Select *ODBC Admin* and then the *System DSN* tab.

   b) Highlight Data Federator, and then click *Configure*.

   c) In the URL option, enter the catalog name and the schema name; for example, `jdbc:leselect://localhost/`**`<catalogname>`**`;schema=`**`<schemaname>`**.
9. For other database types, enter the appropriate information for the selected database type.
10. The *Enable automatic data transfer* check box is selected by default when you create a new datastore and choose `Database` for *Datastore type*. This check box displays for all databases except Attunity Connector, Data Federator, Memory, and Persistent Cache.

Keep Enable automatic data transfer selected to enable transfer tables in this datastore that the Data_Transfer transform can use to push down subsequent database operations.

11. At this point, you can save the datastore or add more information to it:
    - To save the datastore and close the Datastore Editor, click *OK*.
    - To add more information, select *Advanced*.
      To enter values for each configuration option, click the cells under each configuration name.
      For the datastore as a whole, the following buttons and options are available:

Buttons	Description
Import unsupported data types as VARCHAR of size	The data types that the software supports are documented in the Reference Guide. If you want the software to convert a data type in your source that it would not normally support, select this option and enter the number of characters that you will allow.
Edit	Opens the Configurations for Datastore dialog. Use the tool bar on this window to add, configure, and manage multiple configurations for a datastore.
Show ATL	Opens a text window that displays how the software codes the selections you make for this datastore in its scripting language.
OK	Saves selections and closes the Datastore Editor (Create New Datastore) window.
Cancel	Cancels selections and closes the Datastore Editor window.
Apply	Saves selections.

12. Click *OK*.

> Note
>
> On versions of Data Integrator prior to version 11.7.0, the correct database type to use when creating a datastore on Netezza was ODBC. SAP Data Services 11.7.1 provides a specific Netezza option as the *Database type* instead of ODBC. When using Netezza as the database with the software, we recommend that you choose the software's Netezza option as the *Database type* rather than ODBC.

Related Information

Administrator Guide: DSN-less and TNS-less connections [page 30]

Performance Optimization Guide: Data Transfer transform for push-down operations [page 2122]

Reference Guide: Objects, Datastore [page 860]

Creating and managing multiple datastore configurations [page 239]

Ways of importing metadata [page 221]

2.5.2.3 Configuring ODBC data sources on UNIX

To use ODBC data sources on UNIX platforms, you may need to perform additional configuration.

Data Services provides the Connection Manager to simplify configuration of natively-supported ODBC data sources such as MySQL and Teradata. Other ODBC data sources may require manual configuration.

Related Information

Administrator's Guide: Configuring ODBC data sources on UNIX [page 61]

2.5.2.4 Changing a datastore definition

Like all objects, datastores are defined by both options and properties:

- Options control the operation of objects. For example, the name of the database to connect to is a datastore option.
- Properties document the object. For example, the name of the datastore and the date on which it was created are datastore properties. Properties are merely descriptive of the object and do not affect its operation.

2.5.2.4.1 To change datastore options

1. Go to the *Datastores* tab in the object library.
2. Right-click the datastore name and choose *Edit*.

   The Datastore Editor appears (the title bar for this dialog displays Edit Datastore). You can do the following tasks:

   - Change the connection information for the current datastore configuration.
   - Click *Advanced* and change properties for the current configuration.
   - Click *Edit* to add, edit, or delete additional configurations. The Configurations for Datastore dialog opens when you select *Edit* in the Datastore Editor. After you add a new configuration to an existing datastore, you can use the fields in the grid to change connection values and properties for the new configuration.
3. Click *OK*.

   The options take effect immediately.

Related Information

Reference Guide: Database datastores [page 868]

2.5.2.4.2 To change datastore properties

1. Go to the datastore tab in the object library.
2. Right-click the datastore name and select *Properties*.

   The Properties window opens.
3. Change the datastore properties.
4. Click *OK*.

Related Information

Reference Guide: Datastore [page 860]

2.5.2.5 Browsing metadata through a database datastore

The software stores metadata information for all imported objects in a datastore. You can use the software to view metadata for imported or non-imported objects and to check whether the metadata has changed for objects already imported.

2.5.2.5.1 To view imported objects

1. Go to the *Datastores* tab in the object library.
2. Click the plus sign (+) next to the datastore name to view the object types in the datastore. For example, database datastores have functions, tables, and template tables.
3. Click the plus sign (+) next to an object type to view the objects of that type imported from the datastore. For example, click the plus sign (+) next to tables to view the imported tables.

2.5.2.5.2 To sort the list of objects

Click the column heading to sort the objects in each grouping and the groupings in each datastore alphabetically. Click again to sort in reverse-alphabetical order.

2.5.2.5.3 To view datastore metadata

1. Select the *Datastores* tab in the object library.
2. Choose a datastore, right-click, and select *Open*. (Alternatively, you can double-click the datastore icon.)

The software opens the datastore explorer in the workspace. The datastore explorer lists the tables in the datastore. You can view tables in the external database or tables in the internal repository. You can also search through them.

3. Select *External metadata* to view tables in the external database.

   If you select one or more tables, you can right-click for further options.

Command	Description
Open (Only available if you select one table.)	Opens the editor for the table metadata.
Import	Imports (or re-imports) metadata from the database into the repository.
Reconcile	Checks for differences between metadata in the database and metadata in the repository.

4. Select *Repository metadata* to view imported tables.

   If you select one or more tables, you can right-click for further options.

Command	Description
Open (Only available if you select one table)	Opens the editor for the table metadata.
Reconcile	Checks for differences between metadata in the repository and metadata in the database.
Reimport	Reimports metadata from the database into the repository.
Delete	Deletes the table or tables from the repository.
Properties (Only available if you select one table)	Shows the properties of the selected table.
View Data	Opens the View Data window which allows you to see the data currently in the table.

Related Information

To import by searching [page 223]

2.5.2.5.4 To determine if a schema has changed since it was imported

1. In the browser window showing the list of repository tables, select *External Metadata*.

2. Choose the table or tables you want to check for changes.
3. Right-click and choose *Reconcile*.

   The Changed column displays YES to indicate that the database tables differ from the metadata imported into the software. To use the most recent metadata from the software, reimport the table.

   The Imported column displays YES to indicate that the table has been imported into the repository.

2.5.2.5.5 To browse the metadata for an external table

1. In the browser window showing the list of external tables, select the table you want to view.
2. Right-click and choose *Open*.

   A table editor appears in the workspace and displays the schema and attributes of the table.

2.5.2.5.6 To view the metadata for an imported table

1. Select the table name in the list of imported tables.
2. Right-click and select *Open*.

   A table editor appears in the workspace and displays the schema and attributes of the table.

2.5.2.5.7 To view secondary index information for tables

Secondary index information can help you understand the schema of an imported table.

1. From the datastores tab in the Designer, right-click a table to open the shortcut menu.
2. From the shortcut menu, click *Properties* to open the Properties window.
3. In the Properties window, click the *Indexes* tab. The left portion of the window displays the Index list.
4. Click an index to see the contents.

2.5.2.6 Importing metadata through a database datastore

For database datastores, you can import metadata for tables and functions.

2.5.2.6.1 Imported table information

The software determines and stores a specific set of metadata information for tables. After importing metadata, you can edit column names, descriptions, and data types. The edits are propagated to all objects that call these objects.

Metadata	Description
Table name	The name of the table as it appears in the database. Note The maximum length depends on the Data Service repository type. For most repository types the maximum length is 256, for MySQL the length is 64, and for MS SQL server the length is 128.
Table description	The description of the table.
Column name	The name of the column.
Column description	The description of the column.
Column data type	The data type for the column. If a column is defined as an unsupported data type, the software converts the data type to one that is supported. In some cases, if the software cannot convert the data type, it ignores the column entirely.
Column content type	The content type identifies the type of data in the field.
Primary key column	The column(s) that comprise the primary key for the table. After a table has been added to a data flow diagram, these columns are indicated in the column list by a key icon next to the column name.
Table attribute	Information the software records about the table such as the date created and date modified if these values are available.
Owner name	Name of the table owner. Note The owner name for MySQL and Netezza data sources corresponds to the name of the database or schema where the table appears.

Varchar and Column Information from SAP Data Federator tables

Any decimal column imported to Data Serves from an SAP Data Federator data source is converted to the decimal precision and scale(28,6).

Any varchar column imported to the software from an SAP Data Federator data source is varchar(1024).

You may change the decimal precision or scale and varchar size within the software after importing from the SAP Data Federator data source.

2.5.2.6.2 Imported stored function and procedure information

The software can import stored procedures from DB2, MS SQL Server, Oracle, SAP HANA, SQL Anywhere, Sybase ASE, Sybase IQ, and Teredata databases. You can also import stored functions and packages from Oracle. You can use these functions and procedures in the extraction specifications you give Data Services.

Information that is imported for functions includes:

- Function parameters
- Return type
- Name, owner

Imported functions and procedures appear on the *Datastores* tab of the object library. Functions and procedures appear in the *Function* branch of each datastore tree.

You can configure imported functions and procedures through the function wizard and the smart editor in a category identified by the datastore name.

Related Information

Reference Guide: About procedures [page 1697]

2.5.2.6.3 Ways of importing metadata

This section discusses methods you can use to import metadata.

2.5.2.6.3.1 To import by browsing

ℹ Note

Functions cannot be imported by browsing.

1. Open the object library.
2. Go to the *Datastores* tab.
3. Select the datastore you want to use.
4. Right-click and choose *Open*.

   The items available to import through the datastore appear in the workspace.

   In some environments, the tables are organized and displayed as a tree structure. If this is true, there is a plus sign (+) to the left of the name. Click the plus sign to navigate the structure.

   The workspace contains columns that indicate whether the table has already been imported into the software (Imported) and if the table schema has changed since it was imported (Changed). To verify whether the repository contains the most recent metadata for an object, right-click the object and choose Reconcile.
5. Select the items for which you want to import metadata.

   For example, to import a table, you must select a table rather than a folder that contains tables.
6. Right-click and choose *Import*.
7. In the object library, go to the *Datastores* tab to display the list of imported objects.

2.5.2.6.3.2 To import by name

1. Open the object library.
2. Click the *Datastores* tab.
3. Select the datastore you want to use.
4. Right-click and choose *Import By Name*.
5. In the Import By Name window, choose the type of item you want to import from the *Type* list.

   If you are importing a stored procedure, select Function.
6. To import tables:
   a) Enter a table name in the *Name* box to specify a particular table, or select the *All* check box, if available, to specify all tables.
      If the name is case-sensitive in the database (and not all uppercase), enter the name as it appears in the database and use double quotation marks (") around the name to preserve the case.
   b) Enter an owner name in the *Owner* box to limit the specified tables to a particular owner. If you leave the owner name blank, you specify matching tables regardless of owner (that is, any table with the specified table name).
7. To import functions and procedures:
   - In the *Name* box, enter the name of the function or stored procedure.
     If the name is case-sensitive in the database (and not all uppercase), enter the name as it appears in the database and use double quotation marks (") around the name to preserve the case. Otherwise, the software will convert names into all upper-case characters.
     You can also enter the name of a package. An Oracle package is an encapsulated collection of related program objects (e.g., procedures, functions, variables, constants, cursors, and exceptions) stored together in the database. The software allows you to import procedures or functions created within packages and use them as top-level procedures or functions.

If you enter a package name, the software imports all stored procedures and stored functions defined within the Oracle package. You cannot import an individual function or procedure defined within a package.

  - Enter an owner name in the *Owner* box to limit the specified functions to a particular owner. If you leave the owner name blank, you specify matching functions regardless of owner (that is, any function with the specified name).
  - If you are importing an Oracle function or stored procedure and any of the following conditions apply, clear the *Callable from SQL expression* check box. A stored procedure cannot be pushed down to a database inside another SQL statement when the stored procedure contains a DDL statement, ends the current transaction with COMMIT or ROLLBACK, or issues any ALTER SESSION or ALTER SYSTEM commands.

8. Click *OK*.

2.5.2.6.3.3 To import by searching

> ℹ Note
>
> Functions cannot be imported by searching.

1. Open the object library.
2. Click the *Datastores* tab.
3. Select the name of the datastore you want to use.
4. Right-click and select *Search*.

   The Search window appears.
5. Enter the entire item name or some part of it in the *Name* text box.

   If the name is case-sensitive in the database (and not all uppercase), enter the name as it appears in the database and use double quotation marks (") around the name to preserve the case.
6. Select *Contains* or *Equals* from the drop-down list to the right depending on whether you provide a complete or partial search value.

   Equals qualifies only the full search string. That is, you need to search for owner.table_name rather than simply table_name.
7. (Optional) Enter a description in the *Description* text box.
8. Select the object type in the *Type* box.
9. Select the datastore in which you want to search from the *Look In* box.
10. Select *External* from the drop-down box to the right of the *Look In* box.

    External indicates that the software searches for the item in the entire database defined by the datastore.

    Internal indicates that the software searches only the items that have been imported.
11. Go to the *Advanced* tab to search using the software's attribute values.

    The advanced options only apply to searches of imported items.
12. Click *Search*.

    The software lists the tables matching your search criteria.

13. To import a table from the returned list, select the table, right-click, and choose *Import*.

2.5.2.6.4 Reimporting objects

If you have already imported an object such as a datastore, function, or table, you can reimport it, which updates the object's metadata from your database (reimporting overwrites any changes you might have made to the object in the software).

To reimport objects in previous versions of the software, you opened the datastore, viewed the repository metadata, and selected the objects to reimport. In this version of the software, you can reimport objects using the object library at various levels:

- Individual objects — Reimports the metadata for an individual object such as a table or function
- Category node level — Reimports the definitions of all objects of that type in that datastore, for example all tables in the datastore
- Datastore level — Reimports the entire datastore and all its dependent objects including tables, functions, IDOCs, and hierarchies

2.5.2.6.4.1 To reimport objects from the object library

1. In the object library, click the *Datastores* tab.
2. Right-click an individual object and click *Reimport*, or right-click a category node or datastore name and click *Reimport All*.

   You can also select multiple individual objects using Ctrl-click or Shift-click.
3. Click *Yes* to reimport the metadata.
4. If you selected multiple objects to reimport (for example with *Reimport All*), the software requests confirmation for each object unless you check the box *Don't ask me again for the remaining objects*.

   You can skip objects to reimport by clicking No for that object.

   If you are unsure whether to reimport (and thereby overwrite) the object, click View Where Used to display where the object is currently being used in your jobs.

2.5.2.7 Memory datastores

The software also allows you to create a database datastore using *Memory* as the *Database type*. Memory datastores are designed to enhance processing performance of data flows executing in real-time jobs. Data (typically small amounts in a real-time job) is stored in memory to provide immediate access instead of going to the original source data.

A memory datastore is a container for memory tables. A datastore normally provides a connection to a database, application, or adapter. By contrast, a memory datastore contains memory table schemas saved in the repository.

Memory tables are schemas that allow you to cache intermediate data. Memory tables can cache data from relational database tables and hierarchical data files such as XML messages and SAP IDocs (both of which contain nested schemas).

Memory tables can be used to:

- Move data between data flows in real-time jobs. By caching intermediate data, the performance of real-time jobs with multiple data flows is far better than it would be if files or regular tables were used to store intermediate data. For best performance, only use memory tables when processing small quantities of data.
- Store table data in memory for the duration of a job. By storing table data in memory, the LOOKUP_EXT function and other transforms and functions that do not require database operations can access data without having to read it from a remote database.

The lifetime of memory table data is the duration of the job. The data in memory tables cannot be shared between different real-time jobs. Support for the use of memory tables in batch jobs is not available.

2.5.2.7.1 Creating memory datastores

You can create memory datastores using the Datastore Editor window.

2.5.2.7.1.1 To define a memory datastore

1. From the *Project* menu, select *New* > *Datastore* .
2. In the *Name* box, enter the name of the new datastore.

   Be sure to use the naming convention "Memory_DS". Datastore names are appended to table names when table icons appear in the workspace. Memory tables are represented in the workspace with regular table icons. Therefore, label a memory datastore to distinguish its memory tables from regular database tables in the workspace.
3. In the *Datastore type* box keep the default *Database*.
4. In the *Database Type* box select *Memory*.

   No additional attributes are required for the memory datastore.
5. Click *OK*.

2.5.2.7.2 Creating memory tables

When you create a memory table, you do not have to specify the table's schema or import the table's metadata. Instead, the software creates the schema for each memory table automatically based on the preceding schema, which can be either a schema from a relational database table or hierarchical data files such as XML messages. The first time you save the job, the software defines the memory table's schema and saves the table. Subsequently, the table appears with a table icon in the workspace and in the object library under the memory datastore.

2.5.2.7.2.1 To create a memory table

1. From the tool palette, click the template table icon.
2. Click inside a data flow to place the template table.

   The Create Table window opens.
3. From the Create Table window, select the memory datastore.
4. Enter a table name.
5. If you want a system-generated row ID column in the table, click the *Create Row ID* check box.
6. Click *OK*.

   The memory table appears in the workspace as a template table icon.
7. Connect the memory table to the data flow as a target.
8. From the *Project* menu select *Save*.

   In the workspace, the memory table's icon changes to a target table icon and the table appears in the object library under the memory datastore's list of tables.

Related Information

Create Row ID option [page 227]

2.5.2.7.3 Using memory tables as sources and targets

After you create a memory table as a target in one data flow, you can use a memory table as a source or target in any data flow.

Related Information

Real-time Jobs [page 372]

2.5.2.7.3.1 To use a memory table as a source or target

1. In the object library, click the *Datastores* tab.
2. Expand the memory datastore that contains the memory table you want to use.
3. Expand *Tables*.

   A list of tables appears.

4. Select the memory table you want to use as a source or target, and drag it into an open data flow.
5. Connect the memory table as a source or target in the data flow.

   If you are using a memory table as a target, open the memory table's target table editor to set table options.
6. Save the job.

Related Information

Memory table target options [page 227]

2.5.2.7.4 Update Schema option

You might want to quickly update a memory target table's schema if the preceding schema changes. To do this, use the Update Schema option. Otherwise, you would have to add a new memory table to update a schema.

2.5.2.7.4.1 To update the schema of a memory target table

1. Right-click the memory target table's icon in the work space.
2. Select *Update Schema*.

   The schema of the preceding object is used to update the memory target table's schema. The current memory table is updated in your repository. All occurrences of the current memory table are updated with the new schema.

2.5.2.7.5 Memory table target options

The *Delete data from table before loading* option is available for memory table targets. The default is on (the box is selected). To set this option, open the memory target table editor. If you deselect this option, new data will append to the existing table data.

2.5.2.7.6 Create Row ID option

If the *Create Row ID* is checked in the Create Memory Table window, the software generates an integer column called *DI_Row_ID* in which the first row inserted gets a value of 1, the second row inserted gets a value of 2, etc. This new column allows you to use a LOOKUP_EXT expression as an iterator in a script.

> Note
>
> The same functionality is available for other datastore types using the SQL function.

Use the *DI_Row_ID* column to iterate through a table using a `lookup_ext` function in a script. For example:

```
$NumOfRows = total_rows (memory_DS..table1)
$I = 1;
$count=0
while ($count < $NumOfRows)
begin
  $data =
     lookup_ext([memory_DS..table1, 'NO_CACHE','MAX'],[A],[O],[DI_Row_ID,'=',$I]);
  $1 = $I + 1;
  if ($data != NULL)
  begin
  $count = $count + 1;
  end
end
```

In the preceding script, `table1` is a memory table. The table's name is preceded by its datastore name (memory_DS), a dot, a blank space (where a table owner would be for a regular table), then a second dot. There are no owners for memory datastores, so tables are identified by just the datastore name and the table name as shown.

Select the LOOKUP_EXT function arguments (line 7) from the function editor when you define a LOOKUP_EXT function.

The TOTAL_ROWS(DatastoreName.Owner.TableName) function returns the number of rows in a particular table in a datastore. This function can be used with any type of datastore. If used with a memory datastore, use the following syntax: TOTAL_ROWS(**`<DatastoreName..TableName>`**)

The software also provides a built-in function that you can use to explicitly expunge data from a memory table. This provides finer control than the active job has over your data and memory usage. The TRUNCATE_TABLE(**`<DatastoreName..TableName>`**) function can only be used with memory tables.

Related Information

Reference Guide: Functions and Procedures, Descriptions of built-in functions [page 1515]

2.5.2.7.7 Troubleshooting memory tables

- One possible error, particularly when using memory tables, is that the software runs out of virtual memory space. The software exits if it runs out of memory while executing any operation.
- A validation and run time error occurs if the schema of a memory table does not match the schema of the preceding object in the data flow.
  To correct this error, use the Update Schema option or create a new memory table to match the schema of the preceding object in the data flow.
- Two log files contain information specific to memory tables: `trace_memory_reader` log and `trace_memory_loader` log.

2.5.2.8 Persistent cache datastores

The software also allows you to create a database datastore using *Persistent cache* as the *Database type*. Persistent cache datastores provide the following benefits for data flows that process large volumes of data.

- You can store a large amount of data in persistent cache which the software quickly loads into memory to provide immediate access during a job. For example, you can access a lookup table or comparison table locally (instead of reading from a remote database).
- You can create cache tables that multiple data flows can share (unlike a memory table which cannot be shared between different real-time jobs). For example, if a large lookup table used in a lookup_ext function rarely changes, you can create a cache once and subsequent jobs can use this cache instead of creating it each time.

A persistent cache datastore is a container for cache tables. A datastore normally provides a connection to a database, application, or adapter. By contrast, a persistent cache datastore contains cache table schemas saved in the repository.

Persistent cache tables allow you to cache large amounts of data. Persistent cache tables can cache data from relational database tables and files.

> ℹ Note
>
> You cannot cache data from hierarchical data files such as XML messages and SAP IDocs (both of which contain nested schemas). You cannot perform incremental inserts, deletes, or updates on a persistent cache table.

You create a persistent cache table by loading data into the persistent cache target table using one data flow. You can then subsequently read from the cache table in another data flow. When you load data into a persistent cache table, the software always truncates and recreates the table.

2.5.2.8.1 Creating persistent cache datastores

You can create persistent cache datastores using the *Datastore Editor* window.

2.5.2.8.1.1 To define a persistent cache datastore

1. From the *Project* menu, select *New* > *Datastore*.
2. In the *Name* box, enter the name of the new datastore.

   Be sure to use a naming convention such as "Persist_DS". Datastore names are appended to table names when table icons appear in the workspace. Persistent cache tables are represented in the workspace with regular table icons. Therefore, label a persistent cache datastore to distinguish its persistent cache tables from regular database tables in the workspace.
3. In the *Datastore type* box, keep the default *Database*.
4. In the *Database Type* box, select *Persistent cache*.

5. In the *Cache directory* box, you can either type or browse to a directory where you want to store the persistent cache.
6. Click *OK*.

2.5.2.8.2 Creating persistent cache tables

When you create a persistent cache table, you do not have to specify the table's schema or import the table's metadata. Instead, the software creates the schema for each persistent cache table automatically based on the preceding schema. The first time you save the job, the software defines the persistent cache table's schema and saves the table. Subsequently, the table appears with a table icon in the workspace and in the object library under the persistent cache datastore.

You create a persistent cache table in one of the following ways:

- As a target template table in a data flow
- As part of the Data_Transfer transform during the job execution

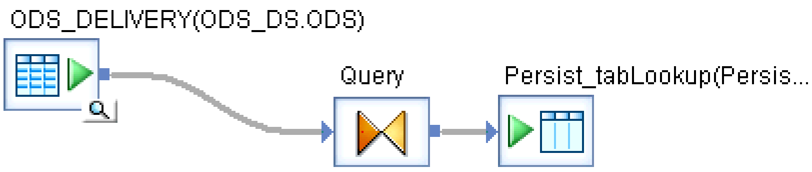
Related Information

Reference Guide: Data_Transfer [page 1071]

2.5.2.8.2.1 To create a persistent cache table as a target in a data flow

1. Use one of the following methods to open the Create Template window:
   - From the tool palette:
     1. Click the template table icon.
     2. Click inside a data flow to place the template table in the workspace.
     3. On the Create Template window, select the persistent cache datastore.
   - From the object library:
     1. Expand a persistent cache datastore.
     2. Click the template table icon and drag it to the workspace.
2. On the Create Template window, enter a table name.
3. Click *OK*.

   The persistent cache table appears in the workspace as a template table icon.
4. Connect the persistent cache table to the data flow as a target (usually a Query transform).



5. In the Query transform, map the Schema In columns that you want to include in the persistent cache table.
6. Open the persistent cache table's target table editor to set table options.
7. On the Options tab of the persistent cache target table editor, you can change the following options for the persistent cache table.
   - *Column comparison* — Specifies how the input columns are mapped to persistent cache table columns. There are two options:
     - Compare_by_position — The software disregards the column names and maps source columns to target columns by position.
     - Compare_by_name — The software maps source columns to target columns by name. This option is the default.
   - *Include duplicate keys* — Select this check box to cache duplicate keys. This option is selected by default.
8. On the Keys tab, specify the key column or columns to use as the key in the persistent cache table.
9. From the *Project* menu select *Save*. In the workspace, the template table's icon changes to a target table icon and the table appears in the object library under the persistent cache datastore's list of tables.

Related Information

Reference Guide:Target persistent cache tables [page 961]

2.5.2.8.3 Using persistent cache tables as sources

After you create a persistent cache table as a target in one data flow, you can use the persistent cache table as a source in any data flow. You can also use it as a lookup table or comparison table.

Related Information

Reference Guide: Persistent cache source [page 951]

2.5.2.9 Linked datastores

Various database vendors support one-way communication paths from one database server to another. Oracle calls these paths database links. In DB2, the one-way communication path from a database server to another database server is provided by an information server that allows a set of servers to get data from remote data sources. In Microsoft SQL Server, linked servers provide the one-way communication path from one database server to another. These solutions allow local users to access data on a remote database, which can be on the local or a remote computer and of the same or different database type.

For example, a local Oracle database server, called Orders, can store a database link to access information in a remote Oracle database, Customers. Users connected to Customers however, cannot use the same link to access data in Orders. Users logged into database Customers must define a separate link, stored in the data dictionary of database Customers, to access data on Orders.

The software refers to communication paths between databases as database links. The datastores in a database link relationship are called linked datastores. The software uses linked datastores to enhance its performance by pushing down operations to a target database using a target datastore.

Related Information

Performance Optimization Guide: Database link support for push-down operations across datastores [page 2124]

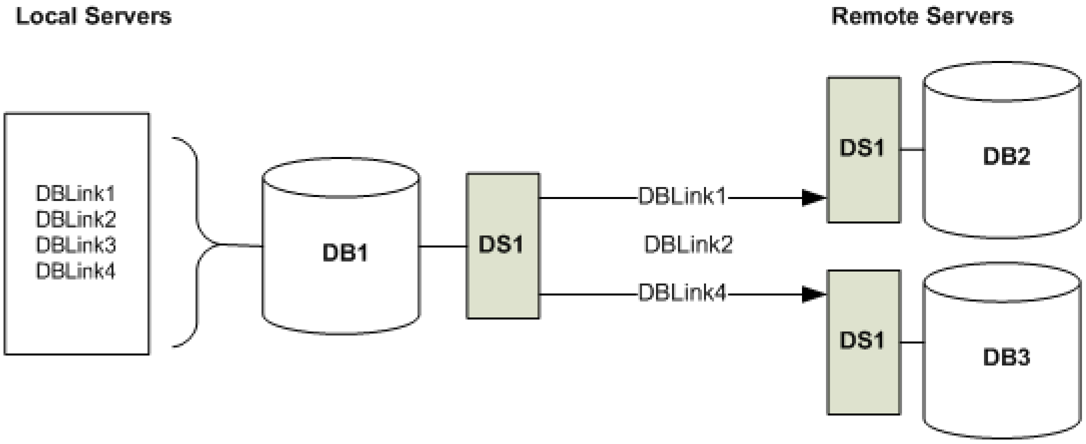
2.5.2.9.1 Relationship between database links and datastores

A database link stores information about how to connect to a remote data source, such as its host name, database name, user name, password, and database type. The same information is stored in an SAP Data Services database datastore.You can associate the datastore to another datastore and then import an external database link as an option of a datastore. The datastores must connect to the databases defined in the database link.

Additional requirements are as follows:

- A local server for database links must be a target server in the software
- A remote server for database links must be a source server in the software
- An external (exists first in a database) database link establishes the relationship between any target datastore and a source datastore
- A Local datastore can be related to zero or multiple datastores using a database link for each remote database
- Two datastores can be related to each other using one link only

The following diagram shows the possible relationships between database links and linked datastores:



Four database links, DBLink 1 through 4, are on database DB1 and the software reads them through datastore Ds1.

- Dblink1 relates datastore Ds1 to datastore Ds2. This relationship is called linked datastore Dblink1 (the linked datastore has the same name as the external database link).
- Dblink2 is not mapped to any datastore in the software because it relates Ds1 with Ds2, which are also related by Dblink1. Although it is not a regular case, you can create multiple external database links that connect to the same remote source. However, the software allows only one database link between a target datastore and a source datastore pair. For example, if you select DBLink1 to link target datastore DS1 with source datastore DS2, you cannot import DBLink2 to do the same.
- Dblink3 is not mapped to any datastore in the software because there is no datastore defined for the remote data source to which the external database link refers.
- Dblink4 relates Ds1 with Ds3.

Related Information

Reference Guide: Datastore editor [page 861]

2.5.3 Adapter datastores

Depending on the adapter implementation, adapters allow you to:

- Browse application metadata
- Import application metadata into a repository
- Move batch and real-time data between the software and applications

SAP offers an Adapter Software Development Kit (SDK) to develop your own custom adapters. Also, you can buy the software pre-packaged adapters to access application metadata and data in any application. For more information on these products, contact your SAP sales representative.

Adapters are represented in Designer by adapter datastores. Jobs provide batch and real-time data movement between the software and applications through an adapter datastore's subordinate objects:

Subordinate Objects	Use as	Used for
Tables	Source or target	Batch data movement
Documents	Source or target	
Functions	Function call in query	
Message functions	Function call in query	Real-time data movement
Outbound messages	Target only	

Adapters can provide access to an application's data and metadata or just metadata. For example, if the data source is SQL-compatible, the adapter might be designed to access metadata, while the software extracts data from or loads data directly to the application.

Related Information

Management Console Guide: Adapters [page 1938]
Source and target objects [page 286]
Real-time source and target objects [page 381]

2.5.3.1 Defining an adapter datastore

You need to define at least one datastore for each adapter through which you are extracting or loading data.

To define a datastore, you must have appropriate access privileges to the application that the adapter serves.

2.5.3.1.1 To define an adapter datastore

To create an adapter datastore, you must first install the adapter on the Job Server computer, configure the Job Server to support local adapters using the Server Manager utility, and ensure that the Job Server's service is running.

1. In the Object Library, click to select the *Datastores* tab.
2. Right-click and select *New*.

   The Create New Datastore window opens.
3. Enter a unique identifying name for the datastore.

   The datastore name appears in the Designer only. It can be the same as the adapter name.
4. In the *Datastore type* list, select *Adapter*.
5. Select a *Job server* from the list.

   Adapters residing on the Job Server computer and registered with the selected Job Server appear in the Job server list.
6. Select an adapter instance from the *Adapter instance name* list.
7. Enter all adapter information required to complete the datastore connection.

   > Note
   >
   > If the developer included a description for each option, the software displays it below the grid. The adapter documentation should list all information required for a datastore connection.

   For the datastore as a whole, the following buttons are available:

Buttons	Description
Edit	Opens the Configurations for Datastore dialog. Use the tool bar on this window to add, configure, and manage multiple configurations for a datastore.

Buttons	Description
Show ATL	Opens a text window that displays how the software will code the selections you make for this datastore in its scripting language.
OK	Saves selections and closes the Datastore Editor (Create New Datastore) window.
Cancel	Cancels selections and closes the Datastore Editor window.
Apply	Saves selections.

8. Click *OK*.

   The datastore configuration is saved in your metadata repository and the new datastore appears in the object library.

After you create a datastore connection, you can browse and/or import metadata from the data source through the adapter.

2.5.3.1.2 To change an adapter datastore's configuration

1. Right-click the datastore you want to browse and select *Edit* to open the Datastore Editor window.
2. Edit configuration information.

   When editing an adapter datastore, enter or select a value. The software looks for the Job Server and adapter instance name you specify. If the Job Server and adapter instance both exist, and the Designer can communicate to get the adapter's properties, then it displays them accordingly. If the Designer cannot get the adapter's properties, then it retains the previous properties.
3. Click *OK*.

   The edited datastore configuration is saved in your metadata repository.

2.5.3.1.3 To delete an adapter datastore and associated metadata objects

1. Right-click the datastore you want to delete and select *Delete*.
2. Click *OK* in the confirmation window.

   The software removes the datastore and all metadata objects contained within that datastore from the metadata repository.

   If these objects exist in established flows, they appear with a deleted icon ⊘.

2.5.3.2 Browsing metadata through an adapter datastore

The metadata you can browse depends on the specific adapter.

2.5.3.2.1 To browse application metadata

1. Right-click the datastore you want to browse and select *Open*.

   A window opens showing source metadata.
2. Scroll to view metadata name and description attributes.
3. Click plus signs [+] to expand objects and view subordinate objects.
4. Right-click any object to check importability.

2.5.3.3 Importing metadata through an adapter datastore

The metadata you can import depends on the specific adapter. After importing metadata, you can edit it. Your edits propagate to all objects that call these objects.

2.5.3.3.1 To import application metadata while browsing

1. Right-click the datastore you want to browse, then select *Open*.
2. Find the metadata object you want to import from the browsable list.
3. Right-click the object and select *Import*.
4. The object is imported into one of the adapter datastore containers (documents, functions, tables, outbound messages, or message functions).

2.5.3.3.2 To import application metadata by name

1. Right-click the datastore from which you want metadata, then select *Import by name*.

   The Import by name window appears containing import parameters with corresponding text boxes.
2. Click each import parameter text box and enter specific information related to the object you want to import.
3. Click *OK*. Any object(s) matching your parameter constraints are imported to one of the corresponding categories specified under the datastore.

2.5.4 Web service datastores

Web service datastores represent a connection from Data Services to an external web service-based data source.

2.5.4.1 Defining a web service datastore

You need to define at least one datastore for each web service with which you are exchanging data.

To define a datastore, you must have the appropriate access priveliges to the web services that the datastore describes.

2.5.4.1.1 To define a web services datastore

1. In the *Datastores* tab of the object library, right-click and select *New*.
2. Enter the name of the new datastore in the *Datastore name* field.

   The name can contain any alphabetical or numeric characters or underscores (_). It cannot contain spaces.
3. Select the *Datastore type*.

   Choose a web service option. When you select a Datastore Type, Data Services displays other options relevant to that type.
4. Specify the *Web Service URL*.

   The URL must accept connections and return the WSDL. For REST web services, you can either enter a URL or the path to the local WADL file. See "Web services technologies" in the *Integrator Guide* for more information about WSDL and WADL files.
5. Click *OK*.
   The datastore configuration is saved in your metadata repository and the new datastore appears in the object library.

After you complete your datastore connection, you can browse and/or import metadata from the web service through the datastore.

2.5.4.1.2 To change a web service datastore's configuration

1. Right-click the datastore you want to browse and select *Edit* to open the Datastore Editor window.
2. Edit configuration information.
3. Click *OK*.
   The edited datastore configuration is saved in your metadata repository.

2.5.4.1.3 To delete a web service datastore and associated metadata objects

1. Right-click the datastore you want to delete and select *Delete*.
2. Click *OK* in the confirmation window.
   Data Services removes the datastore and all metadata objects contained within that datastore from the metadata repository. If these objects exist in established data flows, they appear with a deleted icon.

2.5.4.2 Browsing WSDL and WADL metadata through a web service datastore

Data Services stores metadata information for all imported objects in a datastore. You can use Data Services to view metadata for imported or non-imported objects and to check whether the metadata has changed for objects already imported.

See "Web services technologies" in the *Integrator Guide* for more information about WSDL and WADL files.

2.5.4.2.1 To view imported objects

1. Go to the *Datastores* tab in the object library.
2. Click the plus sign (+) next to the datastore name to view the object types in the datastore. Web service datastores have functions.
3. Click the plus sign (+) next to an object type to view the objects of that type imported from the datastore.

2.5.4.2.2 To sort the list of objects

Click the column heading to sort the objects in each grouping and the groupings in each datastore alphabetically. Click again to sort in reverse-alphabetical order.

2.5.4.2.3 To view WSDL and WADL metadata

1. Select the *Datastores* tab in the object library.
2. Choose a datastore, right-click, and select *Open*. (Alternatively, you can double-click the datastore icon.)
   Data Services opens the datastore explorer in the workspace. The datastore explorer lists the web service ports and operations in the datastore. You can view ports and operations in the external web service or in the internal repository. You can also search through them.
3. Select *External metadata* to view web service ports and operations from the external WSDL or WADL file. See "Web services technologies" in the *Integrator Guide* for more information about WSDL and WADL files.

If you select one or more operations, you can right-click for further options.

Command	Description
Import	Imports (or re-imports) operations from the database into the repository.

4. Select Repository metadata to view imported web service operations.
   If you select one or more operations, you can right-click for further options.

Command	Description
Delete	Deletes the operation or operations from the repository.
Properties	Shows the properties of the selected web service operation.

2.5.4.3 Importing metadata through a web service datastore

For web service datastores, you can import metadata for web service operations.

2.5.4.3.1 To import web service operations

1. Right-click the datastore you want to browse, then select *Open*.
2. Find the web service operation you want to import from the browsable list.
3. Right-click the operation and select *Import*.
   The operation is imported into the web service datastore's function container.

2.5.5 Creating and managing multiple datastore configurations

Creating multiple configurations for a single datastore allows you to consolidate separate datastore connections for similar sources or targets into one source or target datastore with multiple configurations. Then, you can select a set of configurations that includes the sources and targets you want by selecting a system configuration when you execute or schedule the job. The ability to create multiple datastore configurations provides greater ease-of-use for job portability scenarios, such as:

- OEM (different databases for design and distribution)
- Migration (different connections for DEV, TEST, and PROD)
- Multi-instance (databases with different versions or locales)
- Multi-user (databases for central and local repositories)

Related Information

Portability solutions [page 244]

2.5.5.1 Definitions

Refer to the following terms when creating and managing multiple datastore configurations:

Table 4:

Term	Definition
Datastore configuration	Allows you to provide multiple metadata sources or targets for datastores. Each configuration is a property of a datastore that refers to a set of configurable options (such as database connection name, database type, user name, password, and locale) and their values.
Default datastore configuration	The datastore configuration that the software uses for browsing and importing database objects (tables and functions) and executing jobs if no system configuration is specified. If a datastore has more than one configuration, select a default configuration, as needed. If a datastore has only one configuration, the software uses it as the default configuration.
Current datastore configuration	The datastore configuration that the software uses to execute a job. If you define a system configuration, the software will execute the job using the system configuration. Specify a current configuration for each system configuration. If you do not create a system configuration, or the system configuration does not specify a configuration for a datastore, the software uses the default datastore configuration as the current configuration at job execution time.
Database objects	The tables and functions that are imported from a datastore. Database objects usually have owners. Some database objects do not have owners. For example, database objects in an ODBC datastore connecting to an Access database do not have owners.
Owner name	Owner name of a database object (for example, a table) in an underlying database. Also known as database owner name or physical owner name.
Alias	A logical owner name. Create an alias for objects that are in different database environments if you have different owner names in those environments. You can create an alias from the datastore editor for any datastore configuration.
Dependent objects	Dependent objects are the jobs, work flows, data flows, and custom functions in which a database object is used. Dependent object information is generated by the where-used utility.

2.5.5.2 Why use multiple datastore configurations?

By creating multiple datastore configurations, you can decrease end-to-end development time in a multi-source, 24x7, enterprise data warehouse environment because you can easily port jobs among different database types, versions, and instances.

For example, porting can be as simple as:

1. Creating a new configuration within an existing source or target datastore.
2. Adding a datastore alias then map configurations with different object owner names to it.
3. Defining a system configuration then adding datastore configurations required for a particular environment. Select a system configuration when you execute a job.

2.5.5.3 Creating a new configuration

You can create multiple configurations for all datastore types except memory datastores. Use the Datastore Editor to create and edit datastore configurations.

Related Information

Reference Guide: Descriptions of objects, Datastore [page 860]

2.5.5.3.1 To create a new datastore configuration

1. From the Datastores tab of the object library, right-click any existing datastore and select *Edit*.
2. Click *Advanced* to view existing configuration information.

   Each datastore must have at least one configuration. If only one configuration exists, it is the default configuration.

3. Click *Edit* to open the Configurations for Datastore window.
4. Click the *Create New Configuration* icon on the toolbar.

   The Create New Configuration window opens.

5. In the Create New Configuration window:
   a) Enter a unique, logical configuration *Name*.
   b) Select a *Database type* from the drop-down menu.
   c) Select a *Database version* from the drop-down menu.
   d) In the Values for table targets and SQL transforms section, the software pre-selects the *Use values from* value based on the existing database type and version. The Designer automatically uses the existing SQL transform and target values for the same database type and version.

Further, if the database you want to associate with a new configuration is a later version than that associated with other existing configurations, the Designer automatically populates the Use values from with the earlier version.

However, if database type and version are not already specified in an existing configuration, or if the database version is older than your existing configuration, you can choose to use the values from another existing configuration or the default for the database type and version.

e) Select or clear the *Restore values if they already exist* option.

When you delete datastore configurations, the software saves all associated target values and SQL transforms. If you create a new datastore configuration with the same database type and version as the one previously deleted, the Restore values if they already exist option allows you to access and take advantage of the saved value settings.)

- If you keep this option (selected as default) the software uses customized target and SQL transform values from previously deleted datastore configurations.
- If you deselect *Restore values if they already exist*, the software does not attempt to restore target and SQL transform values, allowing you to provide new values.

f) Click *OK* to save the new configuration.

If your datastore contains pre-existing data flows with SQL transforms or target objects, the software must add any new database type and version values to these transform and target objects. Under these circumstances, when you add a new datastore configuration, the software displays the *Added New Values - Modified Objects* window which provides detailed information about affected data flows and modified objects. These same results also display in the Output window of the Designer.

For each datastore, the software requires that one configuration be designated as the default configuration. The software uses the default configuration to import metadata and also preserves the default configuration during export and multi-user operations. Your first datastore configuration is automatically designated as the default; however after adding one or more additional datastore configurations, you can use the datastore editor to flag a different configuration as the default.

When you export a repository, the software preserves all configurations in all datastores including related SQL transform text and target table editor settings. If the datastore you are exporting already exists in the target repository, the software overrides configurations in the target with source configurations. The software exports system configurations separate from other job related objects.

2.5.5.4 Adding a datastore alias

From the datastore editor, you can also create multiple aliases for a datastore then map datastore configurations to each alias.

2.5.5.4.1 To create an alias

1. From within the datastore editor, click *Advanced*, then click *Aliases (Click here to create)*.

   The Create New Alias window opens.
2. Under *Alias Name in Designer*, use only alphanumeric characters and the underscore symbol (_) to enter an alias name.

3. Click *OK*.

   The *Create New Alias* window closes and your new alias appears underneath the Aliases category

When you define a datastore alias, the software substitutes your specified datastore configuration alias for the real owner name when you import metadata for database objects. You can also rename tables and functions after you import them.

Related Information

Renaming table and function owner [page 249]

2.5.5.5 Functions to identify the configuration

The software provides six functions that are useful when working with multiple source and target datastore configurations.

Function	Category	Description
db_type	Miscellaneous	Returns the database type of the current datastore configuration.
db_version	Miscellaneous	Returns the database version of the current datastore configuration.
db_database_name	Miscellaneous	Returns the database name of the current datastore configuration if the database type is MS SQL Server or Sybase ASE.
db_owner	Miscellaneous	Returns the real owner name that corresponds to the given alias name under the current datastore configuration.
current_configuration	Miscellaneous	Returns the name of the datastore configuration that is in use at runtime.
current_system_configuration	Miscellaneous	Returns the name of the current system configuration. If no system configuration is defined, returns a NULL value.

The software links any SQL transform and target table editor settings used in a data flow to datastore configurations. You can also use variable interpolation in SQL text with these functions to enable a SQL transform to perform successfully regardless of which configuration the Job Server uses at job execution time.

Use the Administrator to select a system configuration as well as view the underlying datastore configuration associated with it when you:

- Execute batch jobs

- Schedule batch jobs
- View batch job history
- Create services for real-time jobs

To use multiple configurations successfully, design your jobs so that you do not need to change schemas, data types, functions, variables, and so on when you switch between datastore configurations. For example, if you have a datastore with a configuration for Oracle sources and SQL sources, make sure that the table metadata schemas match exactly. Use the same table names, alias names, number and order of columns, as well as the same column names, data types, and content types.

Related Information

Reference Guide: Descriptions of built-in functions [page 1515]
Reference Guide: SQL [page 1469]
Job portability tips [page 248]

2.5.5.6 Portability solutions

Set multiple source or target configurations for a single datastore if you want to quickly change connections to a different source or target database. The software provides several different solutions for porting jobs.

Related Information

Designer Guide: Multi-user Development [page 798]
Designer Guide: Multi-user Environment Setup [page 802]

2.5.5.6.1 Migration between environments

When you must move repository metadata to another environment (for example from development to test or from test to production) which uses different source and target databases, the process typically includes the following characteristics:

- The environments use the same database type but may have unique database versions or locales.
- Database objects (tables and functions) can belong to different owners.
- Each environment has a unique database connection name, user name, password, other connection properties, and owner mapping.
- You use a typical repository migration procedure. Either you export jobs to an ATL file then import the ATL file to another repository, or you export jobs directly from one repository to another repository.

Because the software overwrites datastore configurations during export, you should add configurations for the target environment (for example, add configurations for the test environment when migrating from development to test) to the source repository (for example, add to the development repository before migrating to the test environment). The Export utility saves additional configurations in the target environment, which means that you do not have to edit datastores before running ported jobs in the target environment.

This solution offers the following advantages:

- Minimal production down time: You can start jobs as soon as you export them.
- Minimal security issues: Testers and operators in production do not need permission to modify repository objects.

Related Information

Administrator Guide: Export/Import [page 117]

2.5.5.6.2 Loading multiple instances

If you must load multiple instances of a data source to a target data warehouse, the task is the same as in a migration scenario except that you are using only one repository.

2.5.5.6.2.1 To load multiple instances of a data source to a target data warehouse

1. Create a datastore that connects to a particular instance.
2. Define the first datastore configuration. This datastore configuration contains all configurable properties such as database type, database connection name, user name, password, database version, and locale information.

   When you define a configuration for an Adapter datastore, make sure that the relevant Job Server is running so the Designer can find all available adapter instances for the datastore.
3. Define a set of alias-to-owner mappings within the datastore configuration. When you use an alias for a configuration, the software imports all objects using the metadata alias rather than using real owner names. This allows you to use database objects for jobs that are transparent to other database instances.
4. Use the database object owner renaming tool to rename owners of any existing database objects.
5. Import database objects and develop jobs using those objects, then run the jobs.
6. To support executing jobs under different instances, add datastore configurations for each additional instance.
7. Map owner names from the new database instance configurations to the aliases that you defined in an earlier step.
8. Run the jobs in all database instances.

Related Information

Renaming table and function owner [page 249]

2.5.5.6.3 OEM deployment

If you design jobs for one database type and deploy those jobs to other database types as an OEM partner, the deployment typically has the following characteristics:

- The instances require various source database types and versions.
- Since a datastore can only access one instance at a time, you may need to trigger functions at run-time to match different instances. If this is the case, the software requires different SQL text for functions (such as lookup_ext and sql) and transforms (such as the SQL transform). The software also requires different settings for the target table (configurable in the target table editor).
- The instances may use different locales.
- Database tables across different databases belong to different owners.
- Each instance has a unique database connection name, user name, password, other connection properties, and owner mappings.
- You export jobs to ATL files for deployment.

2.5.5.6.3.1 To deploy jobs to other database types as an OEM partner

1. Develop jobs for a particular database type following the steps described in the *Loading multiple instances* [page 245] scenario.

   To support a new instance under a new database type, the software copies target table and SQL transform database properties from the previous configuration to each additional configuration when you save it.

   If you selected a bulk loader method for one or more target tables within your job's data flows, and new configurations apply to different database types, open your targets and manually set the bulk loader option (assuming you still want to use the bulk loader method with the new database type). The software does not copy bulk loader options for targets from one database type to another.

   When the software saves a new configuration it also generates a report that provides a list of targets automatically set for bulk loading. Reference this report to make manual changes as needed.
2. If the SQL text in any SQL transform is not applicable for the new database type, modify the SQL text for the new database type.

   If the SQL text contains any hard-coded owner names or database names, consider replacing these names with variables to supply owner names or database names for multiple database types. This way, you will not have to modify the SQL text for each environment.
3. Because the software does not support unique SQL text for each database type or version of the sql(), lookup_ext(), and pushdown_sql() functions, use the db_type() and similar functions to get the database type and version of the current datastore configuration and provide the correct SQL text for that database type and version using the variable substitution (interpolation) technique.

Related Information

Reference Guide: SQL [page 1469]

2.5.5.6.4 Multi-user development

If you are using a central repository management system, allowing multiple developers, each with their own local repository, to check in and check out jobs, the development environment typically has the following characteristics:

- It has a central repository and a number of local repositories.
- Multiple development environments get merged (via central repository operations such as check in and check out) at times. When this occurs, real owner names (used initially to import objects) must be later mapped to a set of aliases shared among all users.
- The software preserves object history (versions and labels).
- The instances share the same database type but may have different versions and locales.
- Database objects may belong to different owners.
- Each instance has a unique database connection name, user name, password, other connection properties, and owner mapping.

In the multi-user development scenario you must define aliases so that the software can properly preserve the history for all objects in the shared environment.

2.5.5.6.4.1 Porting jobs in a multi-user environment

When porting jobs in a multi-user environment, consider these points:

- Rename table owners and function owners to consolidate object database object owner names into aliases.
    - Renaming occurs in local repositories. To rename the database objects stored in the central repository, check out the datastore to a local repository and apply the renaming tool in the local repository.
    - If the objects to be renamed have dependent objects, the software will ask you to check out the dependent objects.
    - If all the dependent objects can be checked out, renaming will create a new object that has the alias and delete the original object that has the original owner name.
    - If all the dependent objects cannot be checked out (data flows are checked out by another user), the software displays a message, which gives you the option to proceed or cancel the operation. If you cannot check out some of the dependent objects, the renaming tool only affects the flows that you can check out. After renaming, the original object will co-exist with the new object. The number of flows affected by the renaming process will affect the Usage and Where-Used information in the Designer for both the original object and the new object.
- You are responsible for checking in all the dependent objects that were checked out during the owner renaming process. Checking in the new objects does not automatically check in the dependent objects that were checked out.
    - The software does not delete original objects from the central repository when you check in the new objects.

  - Use caution because checking in datastores and checking them out as multi-user operations can override datastore configurations.
  - Maintain the datastore configurations of all users by not overriding the configurations they created. Instead, add a configuration and make it your default configuration while working in your own environment.
    - When your group completes the development phase, It is recommended that the last developer delete the configurations that apply to the development environments and add the configurations that apply to the test or production environments.

2.5.5.7 Job portability tips

- The software assumes that the metadata of a table or function is the same across different database types and versions specified in different configurations in the same datastore. For instance, if you import a table when the default configuration of the datastore is Oracle, then later use the table in a job to extract from DB2, your job will run.
- Import metadata for a database object using the default configuration and use that same metadata with all configurations defined in the same datastore.
- The software supports options in some database types or versions that it does not support in others For example, the software supports parallel reading on Oracle hash-partitioned tables, not on DB2 or other database hash-partitioned tables. If you import an Oracle hash-partitioned table and set your data flow to run in parallel, the software will read from each partition in parallel. However, when you run your job using sources from a DB2 environment, parallel reading will not occur.
- The following features support job portability:
  - Enhanced SQL transform
    With the enhanced SQL transform, you can enter different SQL text for different database types/versions and use variable substitution in the SQL text to allow the software to read the correct text for its associated datastore configuration.
  - Enhanced target table editor
    Using enhanced target table editor options, you can configure database table targets for different database types/versions to match their datastore configurations.
  - Enhanced datastore editor
    Using the enhanced datastore editor, when you create a new datastore configuration you can choose to copy the database properties (including the datastore and table target options as well as the SQL transform text) from an existing configuration or use the current values.
- When you design a job that will be run from different database types or versions, name database tables, functions, and stored procedures the same for all sources. If you create configurations for both case-insensitive databases and case-sensitive databases in the same datastore, It is recommended that you name the tables, functions, and stored procedures using all upper-case characters.
- Table schemas should match across the databases in a datastore. This means the number of columns, the column names, and column positions should be exactly the same. The column data types should be the same or compatible. For example, if you have a VARCHAR column in an Oracle source, use a VARCHAR column in the Microsoft SQL Server source too. If you have a DATE column in an Oracle source, use a DATETIME column in the Microsoft SQL Server source. Define primary and foreign keys the same way.
- Stored procedure schemas should match. When you import a stored procedure from one datastore configuration and try to use it for another datastore configuration, the software assumes that the signature of the stored procedure is exactly the same for the two databases. For example, if a stored procedure is a stored

function (only Oracle supports stored functions), then you have to use it as a function with all other configurations in a datastore (in other words, all databases must be Oracle). If your stored procedure has three parameters in one database, it should have exactly three parameters in the other databases. Further, the names, positions, data types, and in/out types of the parameters must match exactly.

Related Information

Advanced Development Guide: Multi-user Development [page 798]
Advanced Development Guide: Multi-user Environment Setup [page 802]

2.5.5.8 Renaming table and function owner

The software allows you to rename the owner of imported tables, template tables, or functions. This process is called owner renaming.

Use owner renaming to assign a single metadata alias instead of the real owner name for database objects in the datastore. Consolidating metadata under a single alias name allows you to access accurate and consistent dependency information at any time while also allowing you to more easily switch between configurations when you move jobs to different environments.

When using objects stored in a central repository, a shared alias makes it easy to track objects checked in by multiple users. If all users of local repositories use the same alias, the software can track dependencies for objects that your team checks in and out of the central repository.

When you rename an owner, the instances of a table or function in a data flow are affected, not the datastore from which they were imported.

2.5.5.8.1 To rename the owner of a table or function

1. From the Datastore tab of the local object library, expand a table, template table, or function category.
2. Right-click the table or function and select *Rename Owner*.
3. Enter a New *Owner Name* then click *Rename*.

   When you enter a New Owner Name, the software uses it as a metadata alias for the table or function.

   > Note
   >
   > If the object you are renaming already exists in the datastore, the software determines if that the two objects have the same schema. If they are the same, then the software proceeds. If they are different, then the software displays a message to that effect. You may need to choose a different object name.

The software supports both case-sensitive and case-insensitive owner renaming.

- If the objects you want to rename are from a case-sensitive database, the owner renaming mechanism preserves case sensitivity.

- If the objects you want to rename are from a datastore that contains both case-sensitive and case-insensitive databases, the software will base the case-sensitivity of new owner names on the case sensitivity of the default configuration. To ensure that all objects are portable across all configurations in this scenario, enter all owner names and object names using uppercase characters.

During the owner renaming process:

- The software updates the dependent objects (jobs, work flows, and data flows that use the renamed object) to use the new owner name.
- The object library shows the entry of the object with the new owner name. Displayed Usage and Where-Used information reflect the number of updated dependent objects.
- If the software successfully updates all the dependent objects, it deletes the metadata for the object with the original owner name from the object library and the repository.

2.5.5.8.2 Using the Rename window in a multi-user scenario

This section provides a detailed description of Rename Owner window behavior in a multi-user scenario.

Using an alias for all objects stored in a central repository allows the software to track all objects checked in by multiple users. If all local repository users use the same alias, the software can track dependencies for objects that your team checks in and out of the central repository.

When you are checking objects in and out of a central repository, depending upon the check-out state of a renamed object and whether that object is associated with any dependent objects, there are several behaviors possible when you select the *Rename* button.

Case 1

Object is not checked out, and object has no dependent objects in the local or central repository.

Behavior: When you click *Rename*, the software renames the object owner.

Case 2

Object is checked out, and object has no dependent objects in the local or central repository.

Behavior: Same as Case 1.

Case 3

Object is not checked out, and object has one or more dependent objects (in the local repository).

Behavior: When you click *Rename*, the software displays a second window listing the dependent objects (that use or refer to the renamed object).

If you click *Continue*, the software renames the objects and modifies the dependent objects to refer to the renamed object using the new owner name. If you click *Cancel*, the Designer returns to the Rename Owner window.

> **Note**
>
> An object might still have one or more dependent objects in the central repository. However, if the object to be renamed is not checked out, the Rename Owner mechanism (by design) does not affect the dependent objects in the central repository.

Case 4

Object is checked out and has one or more dependent objects.

Behavior: This case contains some complexity.

- If you are not connected to the central repository, the status message reads:

  ```
  This object is checked out from central repository <X>. Please select Tools |
  Central Repository… to activate that repository before renaming.
  ```

- If you are connected to the central repository, the Rename Owner window opens.
  When you click Rename, a second window opens to display the dependent objects and a status indicating their check-out state and location. If a dependent object is located in the local repository only, the status message reads:

  ```
  Used only in local repository. No check out necessary.
  ```

- If the dependent object is in the central repository, and it is not checked out, the status message reads:

  ```
  Not checked out
  ```

- If you have the dependent object checked out or it is checked out by another user, the status message shows the name of the checked out repository. For example: `Oracle.production.user1`
  As in Case 2, the purpose of this second window is to show the dependent objects. In addition, this window allows you to check out the necessary dependent objects from the central repository, without having to go to the Central Object Library window.
  Click the *Refresh List* button to update the check out status in the list. This is useful when the software identifies a dependent object in the central repository but another user has it checked out. When that user checks in the dependent object, click *Refresh List* to update the status and verify that the dependent object is no longer checked out.
  To use the Rename Owner feature to its best advantage, check out associated dependent objects from the central repository. This helps avoid having dependent objects that refer to objects with owner names that do not exist. From the central repository, select one or more objects, then right-click and select Check Out.
  After you check out the dependent object, the Designer updates the status. If the check out was successful, the status shows the name of the local repository.

Case 4a

You click *Continue*, but one or more dependent objects are not checked out from the central repository.

In this situation, the software displays another dialog box that warns you about objects not yet checked out and to confirm your desire to continue.

Click *No* to return to the previous dialog box showing the dependent objects. Click *Yes* to proceed with renaming the selected object and to edit its dependent objects. The software modifies objects that are not checked out in the local repository to refer to the new owner name. It is your responsibility to maintain consistency with the objects in the central repository.

Case 4b

You click *Continue*, and all dependent objects are checked out from the central repository.

The software renames the owner of the selected object, and modifies all dependent objects to refer to the new owner name. Although to you, it looks as if the original object has a new owner name, in reality the software has not modified the original object; it created a new object identical to the original, but uses the new owner name. The original object with the old owner name still exists. The software then performs an "undo checkout" on the original object. It becomes your responsibility to check in the renamed object.

When the rename operation is successful, in the *Datastore* tab of the local object library, the software updates the table or function with the new owner name and the Output window displays the following message:

```
Object <Object_Name>: Owner name <Old_Owner> successfully renamed to <New_Owner>,
including references from dependent objects.
```

If the software does not successfully rename the owner, the Output window displays the following message:

```
Object <Object_Name>: Owner name <Old_Owner> could not be renamed to <New_Owner>.
```

2.5.5.9 Defining a system configuration

What is the difference between datastore configurations and system configurations?

Datastore configurations	Each datastore configuration defines a connection to a particular database from a single datastore.
System configurations	Each system configuration defines a set of datastore configurations that you want to use together when running a job. You can define a system configuration if your repository contains at least one datastore with multiple configurations. You can also associate substitution parameter configurations to system configurations.

When designing jobs, determine and create datastore configurations and system configurations depending on your business environment and rules. Create datastore configurations for the datastores in your repository before you create system configurations to organize and associate them.

Select a system configuration to use at run-time. In many enterprises, a job designer defines the required datastore and system configurations and then a system administrator determines which system configuration to use when scheduling or starting a job.

The software maintains system configurations separate from jobs. You cannot check in or check out system configurations in a multi-user environment. However, you can export system configurations to a separate flat file which you can later import.

Related Information

Creating a new configuration [page 241]

2.5.5.9.1 To create a system configuration

1. From the Designer menu bar, select *Tools* ▸ *System Configurations*.

   The *Edit System Configurations* window displays.
2. To add a new system configuration, do one of the following:
   - Click the *Create New Configuration* icon to add a configuration that references the default configuration of the substitution parameters and each datastore connection.
   - Select an existing configuration and click the *Duplicate Configuration* icon to create a copy of the selected configuration.
     You can use the copy as a template and edit the substitution parameter or datastore configuration selections to suit your needs.
3. If desired, rename the new system configuration.
   a) Select the system configuration you want to rename.
   b) Click the *Rename Configuration* icon to enable the edit mode for the configuration name field.
   c) Type a new, unique name and click outside the name field to accept your choice.

   It is recommended that you follow a consistent naming convention and use the prefix `SC_` in each system configuration name so that you can easily identify this file as a system configuration. This practice is particularly helpful when you export the system configuration.
4. From the list, select a substitution parameter configuration to associate with the system configuration.
5. For each datastore, select the datastore configuration you want to use when you run a job using the system configuration.

   If you do not map a datastore configuration to a system configuration, the Job Server uses the default datastore configuration at run-time.
6. Click *OK* to save your system configuration settings.

Related Information

Associating a substitution parameter configuration with a system configuration [page 418]

2.5.5.9.2 To export a system configuration

1. In the object library, select the Datastores tab and right-click a datastore.
2. Select ▶ *Repository* ▶ *Export System Configurations* ◀.

   It is recommended that you add the SC_ prefix to each exported system configuration .atl file to easily identify that file as a system configuration.
3. Click *OK*.

2.6 File Formats

This section discusses file formats, how to use the file format editor, and how to create a file format in the software.

Related Information

Reference Guide: File format [page 920]

2.6.1 Understanding file formats

A file format is a set of properties describing the structure of a flat file (ASCII). File formats describe the metadata structure. A file format describes a specific file. A file format template is a generic description that can be used for multiple data files.

The software can use data stored in files for data sources and targets. A file format defines a connection to a file. Therefore, you use a file format to connect to source or target data when the data is stored in a file rather than a database table. The object library stores file format templates that you use to define specific file formats as sources and targets in data flows.

To work with file formats, perform the following tasks:

- Create a file format template that defines the structure for a file.
- Create a specific source or target file format in a data flow. The source or target file format is based on a template and specifies connection information such as the file name.

File format objects can describe files of the following types:

- Delimited: Characters such as commas or tabs separate each field.
- Fixed width: You specify the column width.
- SAP transport: Use to define data transport objects in SAP application data flows.
- Unstructured text: Use to read one or more files of unstructured text from a directory.
- Unstructured binary: Use to read one or more binary documents from a directory.

Related Information

Supplement for SAP: Connecting to SAP Applications, File formats [page 254]

2.6.2 File format editor

Use the file format editor to set properties for file format templates and source and target file formats. Available properties vary by the mode of the file format editor:

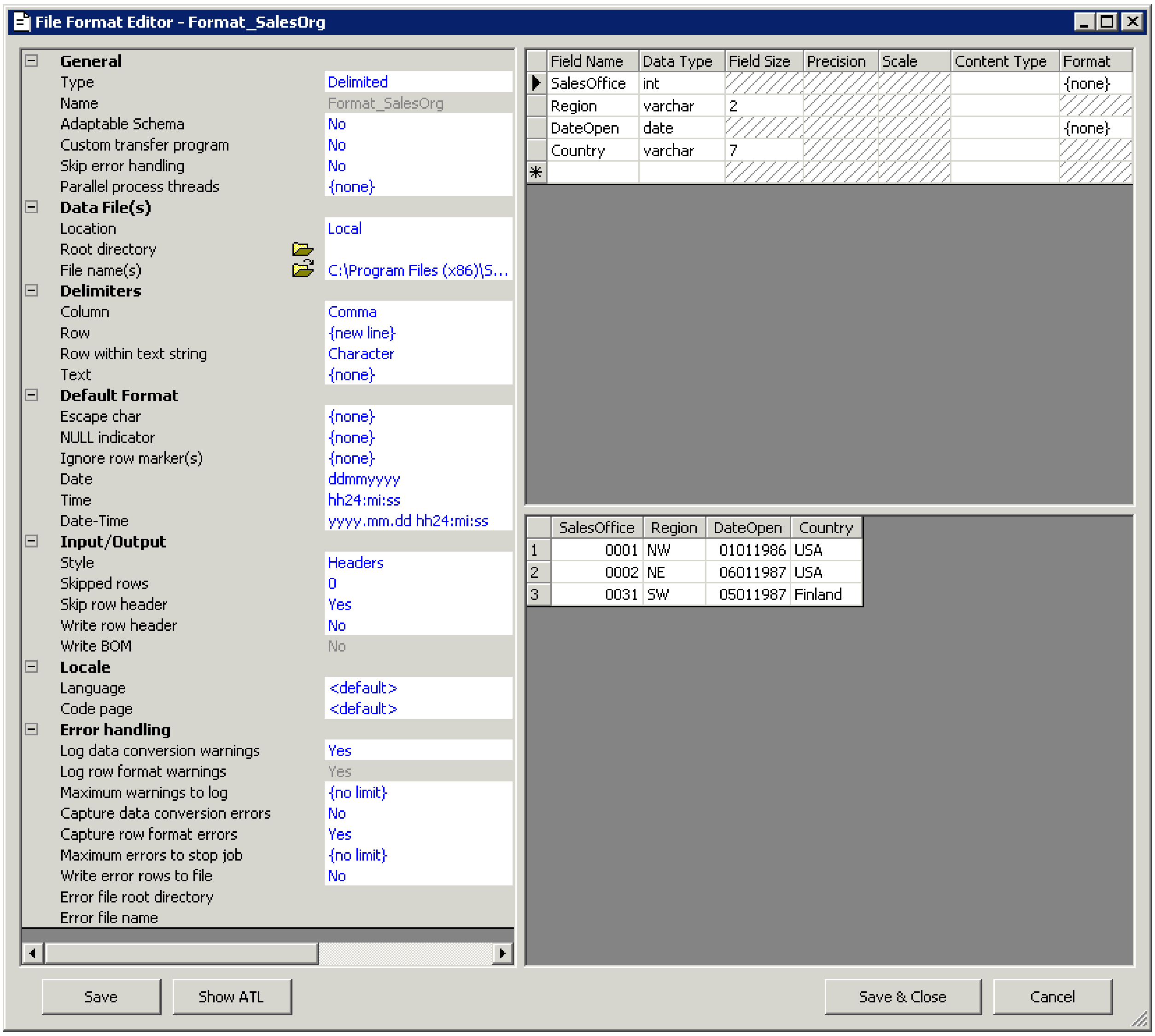
Mode	Description
New mode	Create a new file format template
Edit mode	Edit an existing file format template
Source mode	Edit the file format of a particular source file
Target mode	Edit the file format of a particular target file

The file format editor has three work areas:

Work area	Description
Properties-Values	Edit the values for file format properties. Expand and collapse the property groups by clicking the leading plus or minus.
Column Attributes	Edit and define the columns or fields in the file. Field-specific formats override the default format set in the Properties-Values area.
Data Preview	View how the settings affect sample data.

The file format editor contains "splitter" bars to allow resizing of the window and all the work areas. You can expand the file format editor to the full screen size.

The properties and appearance of the work areas vary with the format of the file.



File Format Editor - Format_SalesOrg

General	
Type	Delimited
Name	Format_SalesOrg
Adaptable Schema	No
Custom transfer program	No
Skip error handling	No
Parallel process threads	{none}
Data File(s)	
Location	Local
Root directory	
File name(s)	C:\Program Files (x86)\S...
Delimiters	
Column	Comma
Row	{new line}
Row within text string	Character
Text	{none}
Default Format	
Escape char	{none}
NULL indicator	{none}
Ignore row marker(s)	{none}
Date	ddmmyyyy
Time	hh24:mi:ss
Date-Time	yyyy.mm.dd hh24:mi:ss
Input/Output	
Style	Headers
Skipped rows	0
Skip row header	Yes
Write row header	No
Write BOM	No
Locale	
Language	<default>
Code page	<default>
Error handling	
Log data conversion warnings	Yes
Log row format warnings	Yes
Maximum warnings to log	{no limit}
Capture data conversion errors	No
Capture row format errors	Yes
Maximum errors to stop job	{no limit}
Write error rows to file	No
Error file root directory	
Error file name	

Field Name	Data Type	Field Size	Precision	Scale	Content Type	Format
SalesOffice	int					{none}
Region	varchar	2				
DateOpen	date					{none}
Country	varchar	7				

	SalesOffice	Region	DateOpen	Country
1	0001	NW	01011986	USA
2	0002	NE	06011987	USA
3	0031	SW	05011987	Finland

You can navigate within the file format editor as follows:

- Switch between work areas using the Tab key.
- Navigate through fields in the Data Preview area with the Page Up, Page Down, and arrow keys.
- Open a drop-down menu in the Properties-Values area by pressing the *ALT*-down arrow key combination.
- When the file format type is fixed-width, you can also edit the column metadata structure in the Data Preview area.

Note

The *Show ATL* button displays a view-only copy of the Transformation Language file generated for your file format. You might be directed to use this by SAP Business User Support.

Related Information

Reference Guide: File format [page 920]

2.6.3 Creating file formats

To specify a source or target file, you create a file format template that defines the structure for a file. When you drag and drop the file format into a data flow; the format represents a file that is based on the template and specifies connection information such as the file name.

2.6.3.1 To create a new file format

1. In the local object library, go to the *Formats* tab, right-click *Flat Files*, and select *New*.
2. For *Type*, select:
   - *Delimited*: For a file that uses a character sequence to separate columns.
   - *Fixed width*: For a file that uses specified widths for each column.
   - *SAP transport*: For data transport objects in SAP application data flows.
   - *Unstructured text*: For one or more files of unstructured text from a directory. The schema is fixed for this type.
   - *Unstructured binary*: For one or more unstructured text and binary documents from a directory. The schema is fixed for this type.

   The options change in the editor based on the type selected.
3. For *Name*, enter a name that describes this file format template.

   After you save this file format template, you cannot change the name.
4. For Delimited and Fixed width files, you can read and load files using a third-party file-transfer program by selecting *Yes* for *Custom transfer program*.
5. Complete the other properties to describe files that this template represents.

   Look for properties available when the file format editor is in source mode or target mode.
6. For source files, some file formats let you specify the structure of the columns in the Column Attributes work area (the upper-right pane):

   a) Enter field name.

   b) Set data types.

   c) Enter field sizes for data types.

   d) Enter scale and precision information for decimal and numeric and data types.

   e) Enter the *Content Type*. If you have added a column while creating a new format, the content type might be provided for you based on the field name. If an appropriate content type is not available, it defaults to blank.

   f) Enter information in the *Format* field for appropriate data types if desired. This information overrides the default format set in the Properties-Values area for that data type.

   You can model a file format on a sample file.

ℹ Note

You do not need to specify columns for files used as targets. If you do specify columns and they do not match the output schema from the preceding transform, the software writes to the target file using the transform's output schema.

ℹ Note

For a decimal or real data type, if you only specify a source column format and the column names and data types in the target schema do not match those in the source schema, the software cannot use the source column format specified. Instead, it defaults to the format used by the code page on the computer where the Job Server is installed.

7. Click *Save & Close* to save the file format template and close the file format editor.

Related Information

Reference Guide: Locales and Multi-byte Functionality [page 1793]
File transfers [page 269]
Reference Guide: File format [page 920]

2.6.3.2 Modeling a file format on a sample file

1. From the *Formats* tab in the local object library, create a new flat file format template or edit an existing flat file format template.
2. Under *Data File(s)*:
   - If the sample file is on your Designer computer, set *Location* to *Local*. Browse to set the *Root directory* and *File(s)* to specify the sample file.

     ℹ Note

     During design, you can specify a file located on the computer where the Designer runs or on the computer where the Job Server runs. Indicate the file location in the Location property. During execution, you must specify a file located on the Job Server computer that will execute the job.

   - If the sample file is on the current Job Server computer, set *Location* to *Job Server*. Enter the *Root directory* and *File(s)* to specify the sample file. When you select *Job Server*, the *Browse* icon is disabled, so you must type the path to the file. You can type an absolute path or a relative path, but the Job Server must be able to access it. For example, a path on UNIX might be `/usr/data/abc.txt`. A path on Windows might be `C:\DATA\abc.txt`.

     ℹ Note

     In the Windows operating system, files are not case-sensitive; however, file names are case sensitive in the UNIX environment. (For example, abc.txt and aBc.txt would be two different files in the same UNIX directory.)

To reduce the risk of typing errors, you can telnet to the Job Server (UNIX or Windows) computer and find the full path name of the file you want to use. Then, copy and paste the path name from the telnet application directly into the Root directory text box in the file format editor. You cannot use the Windows Explorer to determine the exact file location on Windows.

3. If the file type is delimited, set the appropriate column delimiter for the sample file. You can choose from the drop-down list or specify Unicode delimiters by directly typing the Unicode character code in the form of /XXXX, where XXXX is a decimal Unicode character code. For example, /44 is the Unicode character for the comma (,) character.
4. Under *Input/Output*, set *Skip row header* to *Yes* if you want to use the first row in the file to designate field names.

   The file format editor will show the column names in the Data Preview area and create the metadata structure automatically.
5. Edit the metadata structure as needed.

   For both delimited and fixed-width files, you can edit the metadata structure in the Column Attributes work area:

   a) Right-click to insert or delete fields.
   b) Rename fields.
   c) Set data types.
   d) Enter field lengths for the *Blob* and *VarChar* data type.
   e) Enter scale and precision information for *Numeric* and *Decimal* data types.
   f) Enter *Format* field information for appropriate data types, if desired. This format information overrides the default format set in the Properties-Values area for that data type.
   g) Enter the *Content Type* information. You do not need to specify columns for files used as targets. If you have added a column while creating a new format, the content type may auto-fill based on the field name. If an appropriate content type cannot be automatically filled, then it will default to blank.

   For fixed-width files, you can also edit the metadata structure in the Data Preview area:

   a) Click to select and highlight columns.
   b) Right-click to insert or delete fields.

   > Note
   >
   > The Data Preview pane cannot display blob data.
6. Click *Save & Close* to save the file format template and close the file format editor.

2.6.3.3 Replicating and renaming file formats

After you create one file format schema, you can quickly create another file format object with the same schema by replicating the existing file format and renaming it. To save time in creating file format objects, replicate and rename instead of configuring from scratch.

2.6.3.3.1 To create a file format from an existing file format

1. In the Formats tab of the object library, right-click an existing file format and choose *Replicate* from the menu.

   The File Format Editor opens, displaying the schema of the copied file format.
2. Double-click to select the *Name* property value (which contains the same name as the original file format object).
3. Type a new, unique name for the replicated file format.

   > Note
   >
   > You must enter a new name for the replicated file. The software does not allow you to save the replicated file with the same name as the original (or any other existing File Format object). Also, this is your only opportunity to modify the Name property value. Once saved, you cannot modify the name again.
4. Edit other properties as desired.

   Look for properties available when the file format editor is in source mode or target mode.
5. To save and view your new file format schema, click *Save*.

   To terminate the replication process (even after you have changed the name and clicked Save), click Cancel or press the Esc button on your keyboard.
6. Click *Save & Close*.

Related Information

Reference Guide: File format [page 920]

2.6.3.4 To create a file format from an existing flat table schema

1. From the Query editor, right-click a schema and select *Create File format*.

   The File Format editor opens populated with the schema you selected.
2. Edit the new schema as appropriate and click *Save & Close*.

   The software saves the file format in the repository. You can access it from the Formats tab of the object library.

2.6.3.5 To create a specific source or target file

1. Select a flat file format template on the *Formats* tab of the local object library.
2. Drag the file format template to the data flow workspace.

3. Select *Make Source* to define a source file format, or select *Make Target* to define a target file format.
4. Click the name of the file format object in the workspace to open the file format editor.
5. Enter the properties specific to the source or target file.

   Look for properties available when the file format editor is in source mode or target mode.

   Under File name(s), be sure to specify the file name and location in the File and Location properties.

   > Note
   >
   > You can use variables as file names.
6. Connect the file format object to other objects in the data flow as appropriate.

Related Information

Reference Guide: File format [page 920]
Setting file names at run-time using variables [page 413]

2.6.4 Editing file formats

You can modify existing file format templates to match changes in the format or structure of a file. You cannot change the name of a file format template.

For example, if you have a date field in a source or target file that is formatted as mm/dd/yy and the data for this field changes to the format dd-mm-yy due to changes in the program that generates the source file, you can edit the corresponding file format template and change the date format information.

For specific source or target file formats, you can edit properties that uniquely define that source or target such as the file name and location.

> ⚠ Caution
>
> If the template is used in other jobs (usage is greater than 0), changes that you make to the template are also made in the files that use the template.

2.6.4.1 To edit a file format template

1. In the object library *Formats* tab, double-click an existing flat file format (or right-click and choose *Edit)*.

   The file format editor opens with the existing format values.
2. Edit the values as needed.

   Look for properties available when the file format editor is in source mode or target mode.

> ⚠ Caution
>
> If the template is used in other jobs (usage is greater than 0), changes that you make to the template are also made in the files that use the template.

3. Click *Save*.

Related Information

Reference Guide: File format [page 920]

2.6.4.2 To edit a source or target file

1. From the workspace, click the name of a source or target file.

   The file format editor opens, displaying the properties for the selected source or target file.
2. Edit the desired properties.

   Look for properties available when the file format editor is in source mode or target mode.

   To change properties that are not available in source or target mode, you must edit the file's file format template. Any changes you make to values in a source or target file editor override those on the original file format.
3. Click *Save*.

Related Information

Reference Guide: File format [page 920]

2.6.4.3 Change multiple column properties

Use these steps when you are creating a new file format or editing an existing one.

1. Select the *Format* tab in the Object Library.
2. Right-click on an existing file format listed under Flat Files and choose *Edit*.
   The *File Format Editor* opens.
3. In the column attributes area (upper right pane) select the multiple columns that you want to change.
   - To choose a series of columns, select the first column and press the keyboard *Shift* key and select the last column.
   - To choose non-consecutive columns hold down the keyboard *Control* key and select the columns.

4. Right click and choose *Properties*.
   The *Multiple Columns Properties* window opens.
5. Change the Data Type and/or the Content Type and click *OK*.
   The Data Type and Content Type of the selected columns change based on your settings.

2.6.5 File format features

The software offers several capabilities for processing files.

2.6.5.1 Reading multiple files at one time

The software can read multiple files with the same format from a single directory using a single source object.

2.6.5.1.1 To specify multiple files to read

1. Open the editor for your source file format
2. Under *Data File(s)* in the file format editor, set the *Location* of the source files to *Local* or *Job Server*.
3. Set the root directory in *Root directory*.

   > i Note
   >
   > If your Job Server is on a different computer than the Designer, you cannot use Browse to specify the root directory. You must type the path. You can type an absolute path or a relative path, but the Job Server must be able to access it.

4. Under *File name(s)*, enter one of the following:
   - A list of file names separated by commas, or
   - A file name containing a wild card character (* or ?).
     For example:
     1999????.txt might read files from the year 1999
     *.txt reads all files with the txt extension from the specified Root directory

2.6.5.2 Identifying source file names

You might want to identify the source file for each row in your target in the following situations:

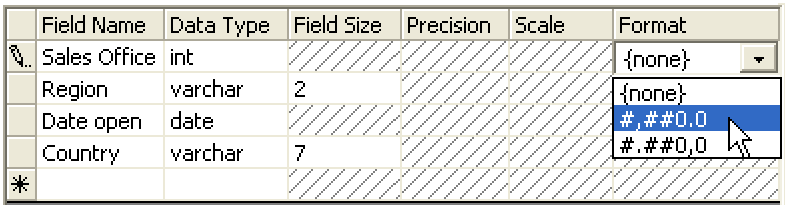
- You specified a wildcard character to read multiple source files at one time
- You load from different source files on different runs

2.6.5.2.1 To identify the source file for each row in the target

1. Under *Source Information* in the file format editor, set *Include file name* to *Yes*. This option generates a column named DI_FILENAME that contains the name of the source file.
2. In the Query editor, map the DI_FILENAME column from Schema In to Schema Out.
3. When you run the job, the DI_FILENAME column for each row in the target contains the source file name.

2.6.5.3 Number formats

The dot (.) and the comma (,) are the two most common formats used to determine decimal and thousand separators for numeric data types. When formatting files in the software, data types in which these symbols can be used include Decimal, Numeric, Float, and Double. You can use either symbol for the thousands indicator and either symbol for the decimal separator. For example: `2,098.65` or `2.089,65`.



	Field Name	Data Type	Field Size	Precision	Scale	Format
✎	Sales Office	int				{none}
	Region	varchar	2			
	Date open	date				
	Country	varchar	7			
*						

Format	Description
{none}	The software expects that the number contains only the decimal separator. The reading of the number data and this decimal separator is determined by Data Service Job Server Locale Region. Comma (,) is the decimal separator when is Data Service Locale is set to a country that uses commas (for example, Germany or France). Dot (.) is the decimal separator when Locale is set to country that uses dots (for example, USA, India, and UK). In this format, the software will return an error if a number contains a thousand separator. When the software writes the data, it only uses the Job Server Locale decimal separator. It does not use thousand separators.
#,##0.0	The software expects that the decimal separator of a number will be a dot (.) and the thousand separator will be a comma (,). When the software loads the data to a flat file, it uses a comma (,) as the thousand separator and a dot (.) as decimal separator.
#.##0,0	The software expects that the decimal separator of a number will be a comma (,) and the thousand separator will be dot (.). When the software loads the data to a flat file, it uses a dot (.) as the thousand separator and comma (,) as decimal separator.

Leading and trailing decimal signs are also supported. For example: `+12,000.00` or `32.32-`.

2.6.5.4 Ignoring rows with specified markers

The file format editor provides a way to ignore rows containing a specified marker (or markers) when reading files. For example, you might want to ignore comment line markers such as `#` and `//`.

Associated with this feature, two special characters — the semicolon (;) and the backslash (\) — make it possible to define multiple markers in your ignore row marker string. Use the semicolon to delimit each marker, and use the backslash to indicate special characters as markers (such as the backslash and the semicolon).

The default marker value is an empty string. When you specify the default value, no rows are ignored.

2.6.5.4.1 To specify markers for rows to ignore

1. Open the file format editor from the Object Library or by opening a source object in the workspace.
2. Find *Ignore row marker(s)* under the *Format* Property.
3. Click in the associated text box and enter a string to indicate one or more markers representing rows that the software should skip during file read and/or metadata creation.

   The following table provides some ignore row marker(s) examples. (Each value is delimited by a semicolon unless the semicolon is preceded by a backslash.)

Marker value(s)	Row(s) ignored
	None (this is the default value)
`abc`	Any that begin with the string `abc`
`abc;def;hi`	Any that begin with `abc` or `def` or `hi`
`abc;\;`	Any that begin with `abc` or `;`
`abc;\\;\;`	Any that begin with `abc` or `\` or `;`

2.6.5.5 Date formats at the field level

You can specify a date format at the field level to overwrite the default date, time, or date-time formats set in the Properties-Values area.

For example, when the *Data Type* is set to Date, you can edit the value in the corresponding *Format* field to a different date format such as:

- yyyy.mm.dd
- mm/dd/yy
- dd.mm.yy

2.6.5.6 Parallel process threads

Data Services can use parallel threads to read and load files to maximize performance.

To specify parallel threads to process your file format:

1. Open the file format editor in one of the following ways:
   - In the Formats tab in the Object Library, right-click a file format name and click *Edit*.
   - In the workspace, double-click the source or target object.
2. Find *Parallel process threads* under the *General* Property.
3. Specify the number of threads to read or load this file format.
   For example, if you have four CPUs on your Job Server computer, enter the number 4 in the *Parallel process threads* box.

Related Information

Performance Optimization Guide: Using Parallel Execution, File multi-threading [page 2150]

2.6.5.7 Error handling for flat-file sources

During job execution, the software processes rows from flat-file sources one at a time. You can configure the File Format Editor to identify rows in flat-file sources that contain the following types of errors:

- Data-type conversion errors — For example, a field might be defined in the File Format Editor as having a data type of integer but the data encountered is actually varchar.
- Row-format errors — For example, in the case of a fixed-width file, the software identifies a row that does not match the expected width value.

These error-handling properties apply to flat-file sources only.

Related Information

Reference Guide: File format [page 920]

2.6.5.7.1 Error-handling options

In the File Format Editor, the *Error Handling* set of properties allows you to choose whether or not to have the software perform the following actions:

- check for either of the two types of flat-file source error
- write the invalid row(s) to a specified error file

- stop processing the source file after reaching a specified number of invalid rows
- log data-type conversion or row-format warnings to the error log; if so, you can limit the number of warnings to log without stopping the job

2.6.5.7.2 About the error file

If enabled, the error file will include both types of errors. The format is a semicolon-delimited text file. You can have multiple input source files for the error file. The file resides on the same computer as the Job Server.

Entries in an error file have the following syntax:

```
source file path and name; row number in source file; Data Services error; column
number where the error occurred; all columns from the invalid row
```

The following entry illustrates a row-format error:

```
d:/acl_work/in_test.txt;2;-80104: 1-3-A column delimiter was seen after column
number <3> for row number <2> in file <d:/acl_work/in_test.txt>. The total number
of columns defined is <3>, so a row delimiter should be seen after column number
<3>. Please check the file for bad data, or redefine the input schema for the file
by editing the file format in the UI.;3;defg;234;def
```

where `3` indicates an error occurred after the third column, and `defg;234;def` are the three columns of data from the invalid row.

> i Note
>
> If you set the file format's *Parallel process thread* option to any value greater than *0* or *{none}*, the row number in source file value will be `-1`.

2.6.5.7.3 Configuring the File Format Editor for error handling

2.6.5.7.3.1 To capture data-type conversion or row-format errors

1. In the object library, click the *Formats* tab.
2. Expand *Flat Files*, right-click a format, and click *Edit*.
3. The File Format Editor opens.
4. To capture data-type conversion errors, under the *Error Handling* properties for *Capture data conversion errors*, click *Yes*.
5. To capture errors in row formats, for *Capture row format errors* click *Yes*.
6. Click *Save* or *Save & Close*.

2.6.5.7.3.2 To write invalid rows to an error file

1. In the object library, click the *Formats* tab.
2. Expand *Flat Files,* right-click a format, and click *Edit*.

   The File Format Editor opens.
3. Under the *Error Handling* properties, click *Yes* for either or both of the *Capture data conversion errors* or *Capture row format errors* properties.
4. For *Write error rows to file*, click *Yes*.

   Two more fields appear: Error file root directory and Error file name.
5. Type an *Error file root directory* in which to store the error file.

   If you type a directory path here, then enter only the file name in the Error file name property.
6. Type an *Error file name*.

   If you leave Error file root directory blank, then type a full path and file name here.
7. Click *Save* or *Save & Close*.

For added flexibility when naming the error file, you can enter a variable that is set to a particular file with full path name. Use variables to specify file names that you cannot otherwise enter such as those that contain multibyte characters

2.6.5.7.3.3 To limit to the number of invalid rows processed before stopping the job

1. In the object library, click the *Formats* tab.
2. Expand *Flat Files*, right-click a format, and click *Edit*.

   The File Format Editor opens.
3. Under the *Error Handling* properties, click *Yes* for either or both the *Capture data conversion errors* or *Capture row format errors* properties.
4. For *Maximum errors to stop job*, type a number.

   > Note
   >
   > This property was previously known as Bad rows limit.
5. Click *Save* or *Save & Close*.

2.6.5.7.3.4 To log data-type conversion warnings in the error log

1. In the object library, click the *Formats* tab.

2. Expand *Flat Files*, right-click a format, and click *Edit*.

   The File Format Editor opens.
3. Under the *Error Handling* properties, for *Log data conversion warnings*, click *Yes*.
4. Click *Save* or *Save & Close*.

2.6.5.7.3.5 To log row-format warnings in the error log

1. In the object library, click the *Formats* tab.
2. Expand *Flat Files*, right-click a format, and click *Edit*.

   The File Format Editor opens.
3. Under the *Error Handling* properties, for *Log row format warnings*, click *Yes*.
4. Click *Save* or *Save & Close*.

2.6.5.7.3.6 To limit to the number of warning messages to log

If you choose to log either data-type or row-format warnings, you can limit the total number of warnings to log without interfering with job execution.

1. In the object library, click the *Formats* tab.
2. Expand *Flat Files*, right-click a format, and click *Edit*.

   The File Format Editor opens.
3. Under the *Error Handling* properties, for *Log row format warnings* or *Log data conversion warnings* (or both),click *Yes*.
4. For *Maximum warnings to log*, type a number.
5. Click *Save* or *Save & Close*.

2.6.6 File transfers

The software can read and load files using a third-party file transfer program for flat files. You can use third-party (custom) transfer programs to:

- Incorporate company-standard file-transfer applications as part of the software job execution
- Provide high flexibility and security for files transferred across a firewall

The custom transfer program option allows you to specify:

- A custom transfer program (invoked during job execution)
- Additional arguments, based on what is available in your program, such as:
  - Connection data

- Encryption/decryption mechanisms
- Compression mechanisms

2.6.6.1 Custom transfer system variables for flat files

When you set custom transfer options for external file sources and targets, some transfer information, like the name of the remote server that the file is being transferred to or from, may need to be entered literally as a transfer program argument. You can enter other information using the following system variables:

Data entered for:	Is substituted for this variable if it is defined in the Arguments field
User name	$AW_USER
Password	$AW_PASSWORD
Local directory	$AW_LOCAL_DIR
File(s)	$AW_FILE_NAME

By using these variables as custom transfer program arguments, you can collect connection information entered in the software and use that data at run-time with your custom transfer program.

For example, the following custom transfer options use a Windows command file (Myftp.cmd) with five arguments. Arguments 1 through 4 are system variables:

- User and Password variables are for the external server
- The Local Directory variable is for the location where the transferred files will be stored in the software
- The File Name variable is for the names of the files to be transferred

Argument 5 provides the literal external server name.

> **Note**
>
> If you do not specify a standard output file (such as `ftp.out` in the example below), the software writes the standard output into the job's trace log.

```
@echo off

set USER=%1
set PASSWORD=%2
set LOCAL_DIR=%3
set FILE_NAME=%4
set LITERAL_HOST_NAME=%5

set INP_FILE=ftp.inp

echo %USER%>%INP_FILE%
echo %PASSWORD%>>%INP_FILE%
echo lcd %LOCAL_DIR%>>%INP_FILE%
echo get %FILE_NAME%>>%INP_FILE%
echo bye>>%INP_FILE%

ftp -s%INPT_FILE% %LITERAL_HOST_NAME%>ftp.out
```

2.6.6.2 Custom transfer options for flat files

Of the custom transfer program options, only the *Program executable* option is mandatory.

Entering *User Name*, *Password,* and *Arguments* values is optional. These options are provided for you to specify arguments that your custom transfer program can process (such as connection data).

You can also use *Arguments* to enable or disable your program's built-in features such as encryption/decryption and compression mechanisms. For example, you might design your transfer program so that when you enter –`sSecureTransportOn` or `-CCompressionYES` security or compression is enabled.

> Note
>
> Available arguments depend on what is included in your custom transfer program. See your custom transfer program documentation for a valid argument list.

You can use the *Arguments* box to enter a user name and password. However, the software also provides separate *User name* and *Password* boxes. By entering the $ **`<AW_USER>`** and $ **`<AW_PASSWORD>`** variables as *Arguments* and then using the *User* and *Password* boxes to enter literal strings, these extra boxes are useful in two ways:

- You can more easily update users and passwords in the software both when you configure the software to use a transfer program and when you later export the job. For example, when you migrate the job to another environment, you might want to change login information without scrolling through other arguments.
- You can use the mask and encryption properties of the *Password* box. Data entered in the *Password* box is masked in log files and on the screen, stored in the repository, and encrypted by Data Services.

  > Note
  >
  > The software sends password data to the custom transfer program in clear text. If you do not allow clear passwords to be exposed as arguments in command-line executables, then set up your custom program to either:

  - Pick up its password from a trusted location.
  - Inherit security privileges from the calling program (in this case, the software).

2.6.6.3 Setting custom transfer options

The custom transfer option allows you to use a third-party program to transfer flat file sources and targets. You can configure your custom transfer program in the File Format Editor window. Like other file format settings, you can override custom transfer program settings if they are changed for a source or target in a particular data flow. You can also edit the custom transfer option when exporting a file format.

2.6.6.3.1 To configure a custom transfer program in the file format editor

1. Select the *Formats* tab in the object library.
2. Right-click *Flat Files* in the tab and select *New*.

   The File Format Editor opens.
3. Select either the *Delimited* or the *Fixed width* file type.

   > Note
   >
   > While the custom transfer program option is not supported by SAP application file types, you can use it as a data transport method for an SAP ABAP data flow.
4. Enter a format name.
5. Select *Yes* for the *Custom transfer program* option.
6. Expand *Custom Transfer* and enter the custom transfer program name and arguments.
7. Complete the other boxes in the file format editor window.

   In the Data File(s) section, specify the location of the file in the software.

   To specify system variables for Root directory and File(s) in the Arguments box:
   - Associate the system variable $ **`<AW_LOCAL_DIR>`** with the local directory argument of your custom transfer program.
   - Associate the system variable $ **`<AW_FILE_NAME>`** with the file name argument of your custom transfer program.

   For example, enter: -l$AW_LOCAL_DIR\$AW_FILE_NAME

   When the program runs, the Root directory and File(s) settings are substituted for these variables and read by the custom transfer program.

   > Note
   >
   > The flag -l used in the example above is a custom program flag. Arguments you can use as custom program arguments in the software depend upon what your custom transfer program expects.
8. Click *Save*.

Related Information

Supplement for SAP: Custom Transfer method [page 2480]
Reference Guide: File format [page 920]

2.6.6.4 Design tips

Keep the following concepts in mind when using the custom transfer options:

- Variables are not supported in file names when invoking a custom transfer program for the file.
- You can only edit custom transfer options in the File Format Editor (or Datastore Editor in the case of SAP application) window before they are exported. You cannot edit updates to file sources and targets at the data flow level when exported. After they are imported, you can adjust custom transfer option settings at the data flow level. They override file format level settings.

When designing a custom transfer program to work with the software, keep in mind that:

- The software expects the called transfer program to return 0 on success and non-zero on failure.
- The software provides trace information before and after the custom transfer program executes. The full transfer program and its arguments with masked password (if any) is written in the trace log. When "Completed Custom transfer" appears in the trace log, the custom transfer program has ended.
- If the custom transfer program finishes successfully (the return code = 0), the software checks the following:
    - For an ABAP data flow, if the transport file does not exist in the local directory, it throws an error and the software stops.
    - For a file source, if the file or files to be read by the software do not exist in the local directory, the software writes a warning message into the trace log.
- If the custom transfer program throws an error or its execution fails (return code is not 0), then the software produces an error with return code and `stdout`/`stderr` output.
- If the custom transfer program succeeds but produces standard output, the software issues a warning, logs the first 1,000 bytes of the output produced, and continues processing.
- The custom transfer program designer must provide valid option arguments to ensure that files are transferred to and from the local directory (specified in the software). This might require that the remote file and directory name be specified as arguments and then sent to the Designer interface using system variables.

Related Information

Supplement for SAP: Custom Transfer method [page 2480]

2.6.7 Creating COBOL copybook file formats

When creating a COBOL copybook format, you can:

- Create just the format, then configure the source after you add the format to a data flow, or
- Create the format and associate it with a data file at the same time.

This section also describes how to:

- Create rules to identify which records represent which schemas using a field ID option.
- Identify the field that contains the length of the schema's record using a record length field option.

Related Information

Reference Guide: Import or Edit COBOL copybook format options [page 850]
Reference Guide: COBOL copybook source options [page 854]
Reference Guide: Data Types, Conversion to or from internal data types [page 1040]

2.6.7.1 To create a new COBOL copybook file format

1. In the local object library, click the *Formats* tab, right-click *COBOL copybooks*, and click *New*.

   The Import COBOL copybook window opens.
2. Name the format by typing a name in the *Format name* field.
3. On the *Format* tab for *File name*, specify the COBOL copybook file format to import, which usually has the extension .cpy.

   During design, you can specify a file in one of the following ways:
   - For a file located on the computer where the Designer runs, you can use the *Browse* button.
   - For a file located on the computer where the Job Server runs, you must type the path to the file. You can type an absolute path or a relative path, but the Job Server must be able to access it.
4. Click *OK*.

   The software adds the COBOL copybook to the object library.
5. The *COBOL Copybook schema name(s)* dialog box displays. If desired, select or double-click a schema name to rename it.
6. Click *OK*.

When you later add the format to a data flow, you can use the options in the source editor to define the source.

Related Information

Reference Guide: COBOL copybook source options [page 854]

2.6.7.2 To create a new COBOL copybook file format and a data file

1. In the local object library, click the *Formats* tab, right-click *COBOL copybooks*, and click *New*.

   The Import COBOL copybook window opens.
2. Name the format by typing a name in the *Format name* field.
3. On the *Format* tab for *File name*, specify to the COBOL copybook file format to import, which usually has the extension .cpy.

During design, you can specify a file in one of the following ways:

- For a file located on the computer where the Designer runs, you can use the Browse button.
- For a file located on the computer where the Job Server runs, you must type the path to the file. You can type an absolute path or a relative path, but the Job Server must be able to access it.

4. Click the *Data File* tab.
5. For *Directory*, type or browse to the directory that contains the COBOL copybook data file to import.

   If you include a directory path here, then enter only the file name in the Name field.
6. Specify the COBOL copybook data file *Name*.

   If you leave Directory blank, then type a full path and file name here.

   During design, you can specify a file in one of the following ways:

   - For a file located on the computer where the Designer runs, you can use the Browse button.
   - For a file located on the computer where the Job Server runs, you must type the path to the file. You can type an absolute path or a relative path, but the Job Server must be able to access it.
7. If the data file is not on the same computer as the Job Server, click the *Data Access* tab. Select *FTP* or *Custom* and enter the criteria for accessing the data file.
8. Click *OK*.
9. The *COBOL Copybook schema name(s)* dialog box displays. If desired, select or double-click a schema name to rename it.
10. Click *OK*.

The Field ID tab allows you to create rules for indentifying which records represent which schemas.

Related Information

Reference Guide: Import or Edit COBOL copybook format options [page 850]

2.6.7.3 To create rules to identify which records represent which schemas

1. In the local object library, click the *Formats* tab, right-click *COBOL copybooks*, and click *Edit*.

   The Edit COBOL Copybook window opens.
2. In the top pane, select a field to represent the schema.
3. Click the *Field ID* tab.
4. On the Field ID tab, select the check box *Use field <schema name.field name> as ID*.
5. Click *Insert below* to add an editable value to the Values list.
6. Type a value for the field.
7. Continue (adding) inserting values as necessary.
8. Select additional fields and insert values as necessary.

9. Click *OK*.

2.6.7.4 To identify the field that contains the length of the schema's record

1. In the local object library, click the *Formats* tab, right-click *COBOL copybooks*, and click *Edit*.

   The Edit COBOL Copybook window opens.
2. Click the *Record Length Field* tab.
3. For the schema to edit, click in its Record Length Field column to enable a drop-down menu.
4. Select the field (one per schema) that contains the record's length.

   The offset value automatically changes to the default of 4; however, you can change it to any other numeric value. The offset is the value that results in the total record length when added to the value in the Record length field.
5. Click *OK*.

2.6.8 Creating Microsoft Excel workbook file formats on UNIX platforms

This section describes how to use a Microsoft Excel workbook as a source with a Job Server on a UNIX platform.

To create Microsoft Excel workbook file formats on Windows, refer to the *Reference Guide*.

To access the workbook, you must create and configure an adapter instance in the Administrator. The following procedure provides an overview of the configuration process. For details about creating adapters, refer to the *Management Console Guide*.

Also consider the following requirements:

- To import the workbook, it must be available on a Windows file system. You can later change the location of the actual file to use for processing in the Excel workbook file format source editor. See the *Reference Guide*.
- To reimport or view data in the Designer, the file must be available on Windows.
- Entries in the error log file might be represented numerically for the date and time fields.
  Additionally, Data Services writes the records with errors to the output (in Windows, these records are ignored).

Related Information

Reference Guide: Excel workbook format [page 914]

Management Console Guide: Adapters [page 1938]

Reference Guide: Excel workbook source options [page 918]

2.6.8.1 To create a Microsoft Excel workbook file format on UNIX

1. Using the Server Manager (**<LINK_DIR>**`/bin/svrcfg`), ensure the UNIX Job Server can support adapters. See the *Installation Guide for UNIX*.
2. Ensure a repository associated with the Job Server is registered in the Central Management Console (CMC). To register a repository in the CMC, see the *Administrator Guide*.
3. In the Administrator, add an adapter to access Excel workbooks. See the *Management Console Guide*.

   You can only configure one Excel adapter per Job Server. Use the following options:

   - On the Status tab, click the job server adapter at right to configure.
   - On the Adapter Configuration tab of Adapter Instances page, click *Add*.
   - On the Adapter Configuration tab, enter the *Adapter instance name*. Type `BOExcelAdapter` (required and case sensitive).
     You may leave all other options at their default values except when processing files larger than 1 MB. In that case, change the Additional Java Launcher Options value to `-Xms64m -Xmx512` or `-Xms128m -Xmx1024m` (the default is `-Xms64m -Xmx256m`). Note that Java memory management can prevent processing very large files (or many smaller files).
4. From the *Administrator* › *Adapter* › *Adapter Instance Status* tab, start the adapter.
5. In the Designer on the *Formats* tab of the object library, create the file format by importing the Excel workbook. For details, see the *Reference Guide*.

Related Information

Administrator Guide: To register a repository in the CMC [page 52]
Management Console Guide: Adding and configuring adapter instances [page 1939]
Reference Guide: Excel workbook format [page 914]

2.6.9 Creating Web log file formats

Web logs are flat files generated by Web servers and are used for business intelligence. Web logs typically track details of Web site hits such as:

- Client domain names or IP addresses
- User names
- Timestamps
- Requested action (might include search string)
- Bytes transferred
- Referred address
- Cookie ID

Web logs use a common file format and an extended common file format.

Common Web log format:

```
151.99.190.27 - - [01/Jan/1997:13:06:51 -0600]
"GET /~bacuslab HTTP/1.0" 301 -4
```

Extended common Web log format:

```
saturn5.cun.com - - [25/JUN/1998:11:19:58 -0500]
"GET /wew/js/mouseover.html HTTP/1.0" 200 1936
"http://av.yahoo.com/bin/query?p=mouse+over+javascript+source+code&hc=0"
"Mozilla/4.02 [en] (x11; U; SunOS 5.6 sun4m)"
```

The software supports both common and extended common Web log formats as sources. The file format editor also supports the following:

- Dash as NULL indicator
- Time zone in date-time, e.g. 01/Jan/1997:13:06:51 –0600

The software includes several functions for processing Web log data:

- Word_ext function
- Concat_data_time function
- WL_GetKeyValue function

Related Information

Word_ext function [page 278]
Concat_date_time function [page 279]
WL_GetKeyValue function [page 279]

2.6.9.1 Word_ext function

The `word_ext` is a string function that extends the word function by returning the word identified by its position in a delimited string. This function is useful for parsing URLs or file names.

Format

```
word_ext(string, word_number, separator(s))
```

A negative word number means count from right to left

Examples

`word_ext('www.bodi.com', 2, '.')` returns '`bodi`'.

`word_ext('www.cs.wisc.edu', -2, '.')` returns '`wisc`'.

`word_ext('www.cs.wisc.edu', 5, '.')` returns `NULL`.

`word_ext('aaa+=bbb+=ccc+zz=dd', 4, '+=')` returns '`zz`'. If 2 separators are specified (+=), the function looks for either one.

`word_ext(',,,,,aaa,,,,bb,,,c ', 2, '.')` returns '`bb`'. This function skips consecutive delimiters.

2.6.9.2 Concat_date_time function

`The concat_date_time` is a date function that returns a datetime from separate date and time inputs.

Format

```
concat_date_time(date, time)
```

Example

```
concat_date_time(MS40."date",MS40."time")
```

2.6.9.3 WL_GetKeyValue function

`The WL_GetKeyValue` is a custom function (written in the Scripting Language) that returns the value of a given keyword. It is useful for parsing search strings.

Format

```
WL_GetKeyValue(string, keyword)
```

Example

`A search in Google for bodi B2B` is recorded in a Web log as:

```
GET "http://www.google.com/search?hl=en&lr=&safe=off&q=bodi+B2B&btnG=Google+Search"
WL_GetKeyValue('http://www.google.com/search?hl=en&lr=&safe=off&q=bodi
+B2B&btnG=Google+Search','q') returns 'bodi+B2B'.
```

2.6.10 Unstructured file formats

Unstructured file formats are a type of flat file format.

To read files that contain unstructured content, create a file format as a source that reads one or more files from a directory. At runtime, the source object in the data flow produces one row per file and contains a reference to each file to access its content. In the data flow, you can use a Text Data Processing transform such as Entity Extraction to process unstructured text or employ another transform to manipulate the data.

The unstructured file format types include:

- *Unstructured text*: Use this format to process a directory of text-based files including
  - text
  - HTML
  - XML

  Data Services stores each file's content using the long data type.
- *Unstructured binary*: Use this format to read binary documents. Data Services stores each file's content using the blob data type.
  - You can process a variety of document formats by obtaining your input from a variety of binary-format files, then passing that blob to the Text Data Processing transform. In this manner, the following formats can be accepted:
    - Microsoft Word: 2003, 2007, and 2010 (Office Open XML)
    - Microsoft PowerPoint: 2003, 2007, and 2010
    - Microsoft Excel: 2003, 2007, and 2010
    - Adobe PDF: 1.3 – 1.7
    - Microsoft RTF: 1.8 and 1.9.1
    - Microsoft Outlook E-mail Message: 2003, 2007, 2010
    - Generic E-mail Message: "`.eml`" files
    - Open Document Text, Spreadsheet, and Presentation: 1.0, 1.1, 1.2
    - Corel WordPerfect: 6.0 (1993) – X5 (2010)
  - You could also use the unstructured binary file format to move a directory of graphic files on disk into a database table. Suppose you want to associate employee photos with the corresponding employee data that is stored in a database. The data flow would include the unstructured binary file format source, a Query transform that associates the employee photo with the employee data using the employee's ID number for example, and the database target table.

Related Information

Reference Guide: Objects, File format [page 920]
Text Data Processing overview [page 318]
Creating file formats [page 257]

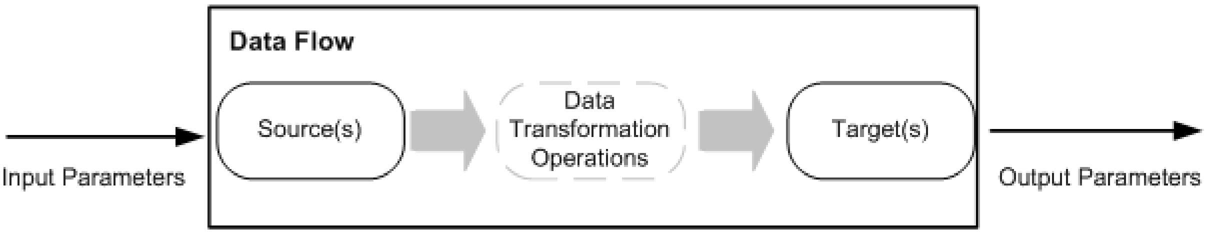
2.7 Data Flows

This section describes the fundamantals of data flows including data flow objects, using lookups, data flow execution, and auditing.

2.7.1 What is a data flow?

Data flows extract, transform, and load data. Everything having to do with data, including reading sources, transforming data, and loading targets, occurs inside a data flow. The lines connecting objects in a data flow represent the flow of data through data transformation steps.

After you define a data flow, you can add it to a job or work flow. From inside a work flow, a data flow can send and receive information to and from other objects through input and output parameters.



2.7.1.1 Naming data flows

Data flow names can include alphanumeric characters and underscores (_). They cannot contain blank spaces.

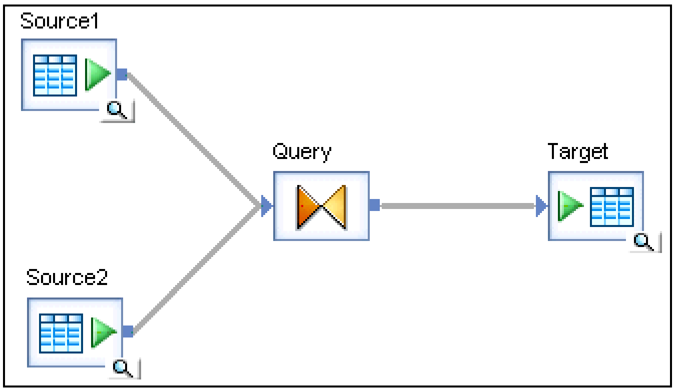
2.7.1.2 Data flow example

Suppose you want to populate the fact table in your data warehouse with new data from two tables in your source transaction database.

Your data flow consists of the following:

- Two source tables
- A join between these tables, defined in a query transform
- A target table where the new rows are placed

You indicate the flow of data through these components by connecting them in the order that data moves through them. The resulting data flow looks like the following:



2.7.1.3 Steps in a data flow

Each icon you place in the data flow diagram becomes a step in the data flow. You can use the following objects as steps in a data flow:

- source
- target
- transforms

The connections you make between the icons determine the order in which the software completes the steps.

Related Information

Source and target objects [page 286]
Transforms [page 303]

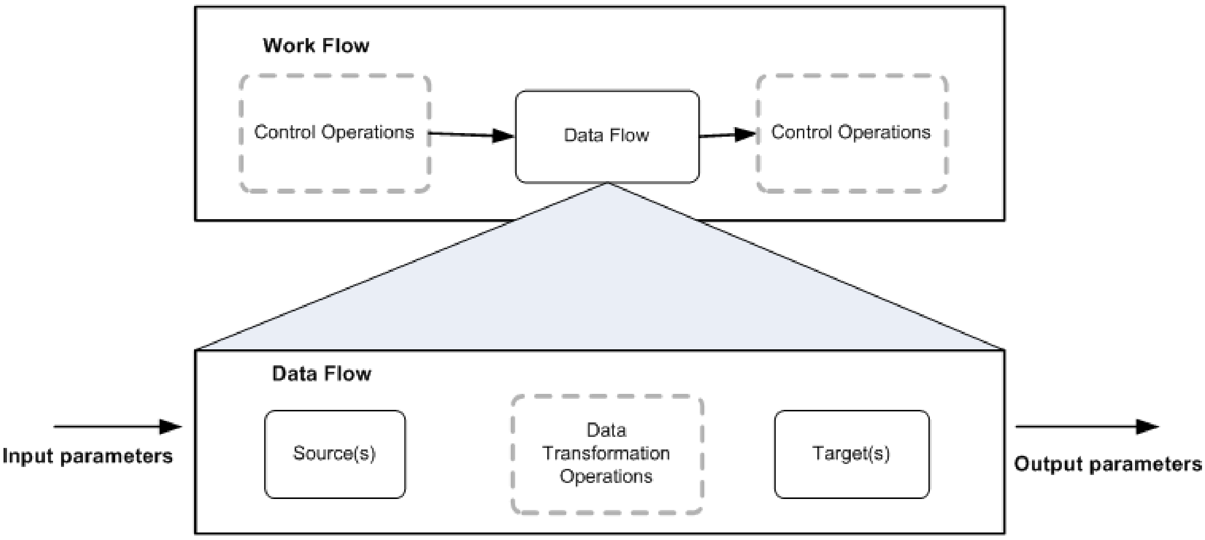
2.7.1.4 Data flows as steps in work flows

Data flows are closed operations, even when they are steps in a work flow. Data sets created within a data flow are not available to other steps in the work flow.

A work flow does not operate on data sets and cannot provide more data to a data flow; however, a work flow can do the following:

- Call data flows to perform data movement operations

- Define the conditions appropriate to run data flows
- Pass parameters to and from data flows



2.7.1.5 Intermediate data sets in a data flow

Each step in a data flow—up to the target definition—produces an intermediate result (for example, the results of a SQL statement containing a WHERE clause), which flows to the next step in the data flow. The intermediate result consists of a set of rows from the previous operation and the schema in which the rows are arranged. This result is called a data set. This data set may, in turn, be further "filtered" and directed into yet another data set.

2.7.1.6 Operation codes

Each row in a data set is flagged with an operation code that identifies the status of the row. The operation codes are as follows:

Operation code	Description
NORMAL	Creates a new row in the target. All rows in a data set are flagged as NORMAL when they are extracted from a source. If a row is flagged as NORMAL when loaded into a target, it is inserted as a new row in the target.
INSERT	Creates a new row in the target. Rows can be flagged as INSERT by transforms in the data flow to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The change is recorded in the target separately from the existing data.

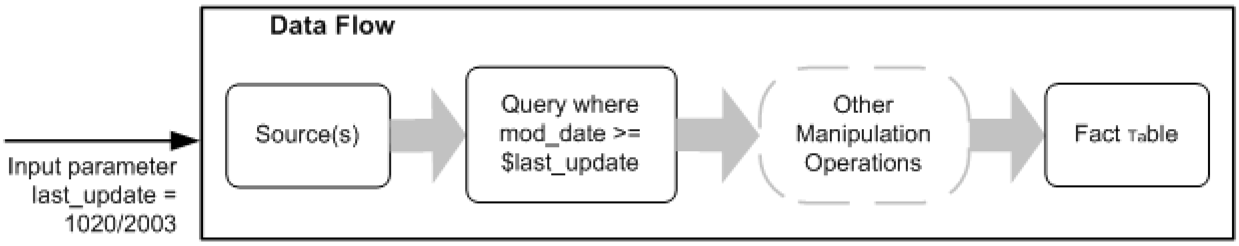
Operation code	Description
DELETE	Is ignored by the target. Rows flagged as DELETE are not loaded. Rows can be flagged as DELETE only by the Map_Operation transform.
UPDATE	Overwrites an existing row in the target. Rows can be flagged as UPDATE by transforms in the data flow to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The change is recorded in the target in the same row as the existing data.

2.7.1.7 Passing parameters to data flows

Data does not flow outside a data flow, not even when you add a data flow to a work flow. You can, however, pass parameters into and out of a data flow. Parameters evaluate single values rather than sets of values.

When a data flow receives parameters, the steps inside the data flow can reference those parameters as variables.

Parameters make data flow definitions more flexible. For example, a parameter can indicate the last time a fact table was updated. You can use this value in a data flow to extract only rows modified since the last update. The following figure shows the parameter `last_update` used in a query to determine the data set used to load the fact table.


Data Flow
Input parameter last_update = 1020/2003
Source(s)
Query where mod_date >= $last_update
Other Manipulation Operations
Fact Table

Related Information

Variables and Parameters [page 398]

2.7.2 Creating and defining data flows

You can create data flows using objects from

- the object library
- the tool palette

After creating a data flow, you can change its properties.

Related Information

To change properties of a data flow [page 285]

2.7.2.1 To define a new data flow using the object library

1. In the object library, go to the *Data Flows* tab.
2. Select the data flow category, right-click and select *New*.
3. Select the new data flow.
4. Drag the data flow into the workspace for a job or a work flow.
5. Add the sources, transforms, and targets you need.

2.7.2.2 To define a new data flow using the tool palette

1. Select the data flow icon in the tool palette.
2. Click the workspace for a job or work flow to place the data flow.

   You can add data flows to batch and real-time jobs. When you drag a data flow icon into a job, you are telling the software to validate these objects according the requirements of the job type (either batch or real-time).
3. Add the sources, transforms, and targets you need.

2.7.2.3 To change properties of a data flow

1. Right-click the data flow and select *Properties*.

   The Properties window opens for the data flow.
2. Change desired properties of a data flow.
3. Click *OK*.

   This table describes the various properties you can set for the data flow.

Table 5:

Option	Description
Execute only once	When you specify that a data flow should only execute once, a batch job will never re-execute that data flow after the data flow completes successfully, except if the data flow is contained in a work flow that is a recovery unit that re-executes and has not completed successfully elsewhere outside the re-

Option	Description
	covery unit. It is recommended that you do not mark a data flow as Execute only once if a parent work flow is a recovery unit.
Use database links	Database links are communication paths between one database server and another. Database links allow local users to access data on a remote database, which can be on the local or a remote computer of the same or different database type.
Degree of parallelism	Degree Of Parallelism (DOP) is a property of a data flow that defines how many times each transform within a data flow replicates to process a parallel subset of data.
Cache type	You can cache data to improve performance of operations such as joins, groups, sorts, filtering, lookups, and table comparisons. You can select one of the following values for the Cache type option on your data flow Properties window: ○ *In-Memory*: Choose this value if your data flow processes a small amount of data that can fit in the available memory. ○ *Pageable*: This value is the default.

Related Information

Performance Optimization Guide: Maximizing Push-Down Operations, Database link support for push-down operations across datastores [page 2124]
Performance Optimization Guide: Using parallel Execution, Degree of parallelism [page 2142]
Performance Optimization Guide: Using Caches [page 2129]
Reference Guide: Objects, Data flow [page 857]

2.7.3 Source and target objects

A data flow directly reads and loads data using two types of objects:

Source objects—Define sources from which you read data.

Target objects—Define targets to which you write (or load) data.

Related Information

Source objects [page 287]
Target objects [page 287]

2.7.3.1 Source objects

Source objects represent data sources read from data flows.

Source object	Description	Software access
Table	A file formatted with columns and rows as used in relational databases.	Direct or through adapter
Template table	A template table that has been created and saved in another data flow (used in development).	Direct
File	A delimited or fixed-width flat file.	Direct
Document	A file with an application- specific format (not readable by SQL or XML parser).	Through adapter
XML file	A file formatted with XML tags.	Direct
XML message	Used as a source in real-time jobs.	Direct

You can also use IDoc messages as real-time sources for SAP applications.

Related Information

Template tables [page 289]
Real-time source and target objects [page 381]
Supplement for SAP: IDoc sources in real-time jobs [page 2496]

2.7.3.2 Target objects

Target objects represent data targets that can be written to in data flows.

Target object	Description	Software access
Table	A file formatted with columns and rows as used in relational databases.	Direct or through adapter
Template table	A table whose format is based on the output of the preceding transform (used in development).	Direct
File	A delimited or fixed-width flat file.	Direct
Document	A file with an application-specific format (not readable by SQL or XML parser).	Through adapter
XML file	A file formatted with XML tags.	Direct
XML template file	An XML file whose format is based on the preceding transform output (used in development, primarily for debugging data flows).	Direct

Target object	Description	Software access
XML message	See *Real-time source and target objects* [page 381].	
Outbound message	See *Real-time source and target objects* [page 381].	

You can also use IDoc messages as real-time sources for SAP applications.

Related Information

Supplement for SAP: IDoc targets in real-time jobs [page 2499]

2.7.3.3 Adding source or target objects to data flows

Fulfill the following prerequisites before using a source or target object in a data flow:

For	Prerequisite
Tables accessed directly from a database	Define a database datastore and import table metadata.
Template tables	Define a database datastore.
Files	Define a file format and import the file.
XML files and messages	Import an XML file format.
Objects accessed through an adapter	Define an adapter datastore and import object metadata.

Related Information

Database datastores [page 210]
Template tables [page 289]
File Formats [page 254]
To import a DTD or XML Schema format [page 354]
Adapter datastores [page 233]

2.7.3.3.1 To add a source or target object to a data flow

1. Open the data flow in which you want to place the object.
2. If the object library is not already open, select *Tools* › *Object Library* to open it.
3. Select the appropriate object library tab: Choose the Formats tab for flat files, DTDs, or XML Schemas, or choose the Datastores tab for database and adapter objects.
4. Select the object you want to add as a source or target. (Expand collapsed lists by clicking the plus sign next to a container icon.)

   For a new template table, select the Template Table icon from the tool palette.

   For a new XML template file, select the Template XML icon from the tool palette.
5. Drop the object in the workspace.
6. For objects that can be either sources or targets, when you release the cursor, a popup menu appears. Select the kind of object to make.

   For new template tables and XML template files, when you release the cursor, a secondary window appears. Enter the requested information for the new template object. Names can include alphanumeric characters and underscores (_). Template tables cannot have the same name as an existing table within a datastore.
7. The source or target object appears in the workspace.
8. Click the object name in the workspace

   The software opens the editor for the object. Set the options you require for the object.

> i Note
>
> Ensure that any files that reference flat file, DTD, or XML Schema formats are accessible from the Job Server where the job will be run and specify the file location relative to this computer.

2.7.3.4 Template tables

During the initial design of an application, you might find it convenient to use template tables to represent database tables. With template tables, you do not have to initially create a new table in your DBMS and import the metadata into the software. Instead, the software automatically creates the table in the database with the schema defined by the data flow when you execute a job.

After creating a template table as a target in one data flow, you can use it as a source in other data flows. Though a template table can be used as a source table in multiple data flows, it can only be used as a target in one data flow.

Template tables are particularly useful in early application development when you are designing and testing a project. If you modify and save the data transformation operation in the data flow where the template table is a target, the schema of the template table automatically changes. Any updates to the schema are automatically made to any other instances of the template table. During the validation process, the software warns you of any errors such as those resulting from changing the schema.

2.7.3.4.1 To create a target template table

1. Use one of the following methods to open the *Create Template* window:
    - From the tool palette:
        - Click the template table icon.
        - Click inside a data flow to place the template table in the workspace.
        - On the Create Template window, select a datastore.
    - From the object library:
        - Expand a datastore.
        - Click the template table icon and drag it to the workspace.
    - From the object library:
        - Expand a datastore.
        - Click the template table icon and drag it to the workspace.
2. On the Create Template window, enter a table name.
3. Click *OK*.

    The table appears in the workspace as a template table icon.
4. Connect the template table to the data flow as a target (usually a Query transform).
5. In the Query transform, map the Schema In columns that you want to include in the target table.
6. From the *Project* menu select *Save*.

    In the workspace, the template table's icon changes to a target table icon and the table appears in the object library under the datastore's list of tables.

After you are satisfied with the design of your data flow, save it. When the job is executed, software uses the template table to create a new table in the database you specified when you created the template table. Once a template table is created in the database, you can convert the template table in the repository to a regular table.

2.7.3.5 Converting template tables to regular tables

You must convert template tables to regular tables to take advantage of some features such as bulk loading. Other features, such as exporting an object, are available for template tables.

> **i Note**
>
> Once a template table is converted, you can no longer alter the schema.

2.7.3.5.1 To convert a template table into a regular table from the object library

1. Open the object library and go to the *Datastores* tab.

2. Click the plus sign (+) next to the datastore that contains the template table you want to convert.

   A list of objects appears.
3. Click the plus sign (+) next to *Template Tables*.

   The list of template tables appears.
4. Right-click a template table you want to convert and select *Import Table*.

   The software converts the template table in the repository into a regular table by importing it from the database. To update the icon in all data flows, choose *View* > *Refresh* . In the datastore object library, the table is now listed under Tables rather than Template Tables.

2.7.3.5.2 To convert a template table into a regular table from a data flow

1. Open the data flow containing the template table.
2. Right-click on the template table you want to convert and select *Import Table*.

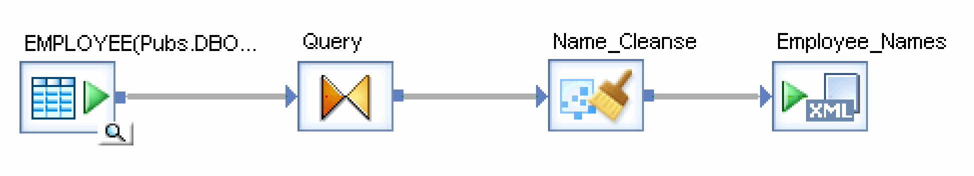
After a template table is converted into a regular table, you can no longer change the table's schema.

2.7.4 Adding columns within a data flow

Within a data flow, the *Propagate Column From* command adds an existing column from an upstream source or transform through intermediate objects to the selected endpoint. Columns are added in each object with no change to the data type or other attributes. When there is more than one possible path between the starting point and ending point, you can specify the route for the added columns.

Column propagation is a pull-through operation. The *Propagate Column From* command is issued from the object where the column is needed. The column is pulled from the selected upstream source or transform and added to each of the intermediate objects as well as the selected endpoint object.

For example, in the data flow below, the `Employee` source table contains employee name information as well as employee ID, job information, and hire dates. The `Name_Cleanse` transform is used to standardize the employee names. Lastly, the data is output to an XML file called `Employee_Names`.



After viewing the output in the `Employee_Names` table, you realize that the middle initial (`minit` column) should be included in the output. You right-click the top-level schema of the `Employee_Names` table and select *Propagate Column From*. The *Propagate Column to Employee_Names* window appears.

In the left pane of the *Propagate Column to Employee_Names* window, select the *Employee* source table from the list of objects. The list of output columns displayed in the right pane changes to display the columns in the schema of the selected object. Select the *MINIT* column as the column you want to pull through from the source, and then click *Propagate*.

The `minit` column schema is carried through the `Query` and `Name_Cleanse` transforms to the `Employee_Names` table.

Characteristics of propagated columns are as follows:

- The *Propagate Column From* command can be issued from the top-level schema of either a transform or a target.
- Columns are added in each object with no change to the data type or other attributes. Once a column is added to the schema of an object, the column functions in exactly the same way as if it had been created manually.
- The propagated column is added at the end of the schema list in each object.
- The output column name is auto-generated to avoid naming conflicts with existing columns. You can edit the column name, if desired.
- Only columns included in top-level schemas can be propagated. Columns in nested schemas cannot be propagated.
- A column can be propagated more than once. Any existing columns are shown in the right pane of the *Propagate Column to* window in the *Already Exists In* field. Each additional column will have a unique name.
- Multiple columns can be selected and propagated in the same operation.

> Note
>
> You cannot propagate a column through a Hierarchy_Flattening transform or a Table_Comparison transform.

2.7.4.1 To add columns within a data flow

Within a data flow, the *Propagate Column From* command adds an existing column from an upstream source or transform through intermediate objects to a selected endpoint. Columns are added in each object with no change to the data type or other attributes.

To add columns within a data flow:

1. In the downstream object where you want to add the column (the endpoint), right-click the top-level schema and click *Propagate Column From*.

   The *Propagate Column From* can be issued from the top-level schema in a transform or target object.
2. In the left pane of the *Propagate Column to* window, select the upstream object that contains the column you want to map.

   The available columns in that object are displayed in the right pane along with a list of any existing mappings from that column.
3. In the right pane, select the column you wish to add and click either *Propagate* or *Propagate and Close*.
   One of the following occurs:

   - If there is a single possible route, the selected column is added through the intermediate transforms to the downstream object.
   - If there is more than one possible path through intermediate objects, the *Choose Route to* dialog displays. This may occur when your data flow contains a Query transform with multiple input objects. Select the path you prefer and click *OK*.

2.7.4.2 Propagating columns in a data flow containing a Merge transform

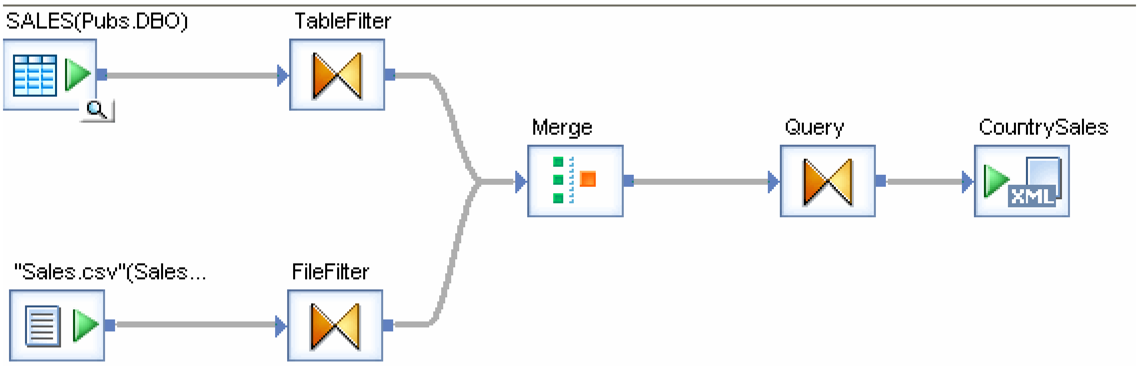
In valid data flows that contain two or more sources which are merged using a Merge transform, the schema of the inputs into the Merge transform must be identical. All sources must have the same schema, including:

- the same number of columns
- the same column names
- like columns must have the same data type

In order to maintain a valid data flow when propagating a column through a Merge transform, you must make sure to meet this restriction.

When you propagate a column and a Merge transform falls between the starting point and ending point, a message warns you that after the propagate operation completes the data flow will be invalid because the input schemas in the Merge transform will not be identical. If you choose to continue with the column propagation operation, you must later add columns to the input schemas in the Merge transform so that the data flow is valid.

For example, in the data flow shown below, the data from each source table is filtered and then the results are merged in the Merge transform.



If you choose to propagate a column from the `SALES(Pubs.DBO)` source to the `CountrySales` target, the column would be added to the `TableFilter` schema but not to the `FileFilter` schema, resulting in differing input schemas in the Merge transform and an invalid data flow.

In order to maintain a valid data flow, when propagating a column through a Merge transform you may want to follow a multi-step process:

1. Ensure that the column you want to propagate is available in the schemas of all the objects that lead into the Merge transform on the upstream side. This ensures that all inputs to the Merge transform are identical and the data flow is valid.
2. Propagate the column on the downstream side of the Merge transform to the desired endpoint.

2.7.5 Lookup tables and the lookup_ext function

Lookup tables contain data that other tables reference. Typically, lookup tables can have the following kinds of columns:

- Lookup column—Use to match a row(s) based on the input values. You apply operators such as =, >, <, ~ to identify a match in a row. A lookup table can contain more than one lookup column.
- Output column—The column returned from the row that matches the lookup condition defined for the lookup column. A lookup table can contain more than one output column.
- Return policy column—Use to specify the data to return in the case where multiple rows match the lookup condition(s).

Use the lookup_ext function to retrieve data from a lookup table based on user-defined lookup conditions that match input data to the lookup table data. Not only can the lookup_ext function retrieve a value in a table or file based on the values in a different source table or file, but it also provides extended functionality that lets you do the following:

- Return multiple columns from a single lookup
- Choose from more operators, including pattern matching, to specify a lookup condition
- Specify a return policy for your lookup
- Call lookup_ext in scripts and custom functions (which also lets you reuse the lookup(s) packaged inside scripts)
- Define custom SQL using the SQL_override parameter to populate the lookup cache, which is useful for narrowing large quantities of data to only the sections relevant for your lookup(s)
- Call lookup_ext using the function wizard in the query output mapping to return multiple columns in a Query transform
- Choose a caching strategy, for example decide to cache the whole lookup table in memory or dynamically generate SQL for each input record
- Use lookup_ext with memory datastore tables or persistent cache tables. The benefits of using persistent cache over memory tables for lookup tables are:
  - Multiple data flows can use the same lookup table that exists on persistent cache.
  - The software does not need to construct the lookup table each time a data flow uses it.
  - Persistent cache has no memory constraints because it is stored on disk and the software quickly pages it into memory.
- Use pageable cache (which is not available for the lookup and lookup_seq functions)
- Use expressions in lookup tables and return the resulting values

For a description of the related functions lookup and lookup_seq, see the *Reference Guide*.

Related Information

Reference Guide: Functions and Procedures, lookup_ext [page 1605]
Performance Optimization Guide: Using Caches, Caching data [page 2129]

2.7.5.1 Accessing the lookup_ext editor

Lookup_ext has its own graphic editor. You can invoke the editor in two ways:

- Add a new function call inside a Query transform—Use this option if you want the lookup table to return more than one column.

- From the Mapping tab in a query or script function.

2.7.5.1.1 To add a new function call

1. In the Query transform *Schema out* pane, without selecting a specific output column right-click in the pane and select *New Function Call*.
2. Select the Function category *Lookup Functions* and the Function name *lookup_ext*.
3. Click *Next* to invoke the editor.

In the Output section, you can add multiple columns to the output schema.

An advantage of using the new function call is that after you close the lookup_ext function window, you can reopen the graphical editor to make modifications (right-click the function name in the schema and select *Modify Function Call*).

2.7.5.1.2 To invoke the lookup_ext editor from the Mapping tab

1. Select the output column name.
2. On the *Mapping* tab, click *Functions*.
3. Select the *Function category Lookup Functions* and the *Function name lookup_ext*.
4. Click *Next* to invoke the editor.

In the Output section, *Variable* replaces *Output column name*. You can define one output column that will populate the selected column in the output schema. When lookup_ext returns more than one output column, use variables to store the output values, or use lookup_ext as a new function call as previously described in this section.

With functions used in mappings, the graphical editor isn't available, but you can edit the text on the *Mapping* tab manually.

2.7.5.2 Example: Defining a simple lookup_ext function

This procedure describes the process for defining a simple lookup_ext function using a new function call. The associated example illustrates how to use a lookup table to retrieve department names for employees.

For details on all the available options for the lookup_ext function, see the *Reference Guide*.

1. In a data flow, open the Query editor.
2. From the *Schema in* pane, drag the ID column to the *Schema out* pane.
3. Select the ID column in the *Schema out* pane, right-click, and click *New Function Call*. Click *Insert Below*.
4. Select the *Function category Lookup Functions* and the *Function name lookup_ext* and click *Next*.
   The lookup_ext editor opens.
5. In the *Lookup_ext - Select Parameters* window, select a lookup table:

a) Next to the *Lookup table* text box, click the drop-down arrow and double-click the datastore, file format, or current schema that includes the table.

b) Select the lookup table and click *OK*.

In the example, the lookup table is a file format called ID_lookup.txt that is in D:\Data.

6. For the *Cache spec*, the default of PRE_LOAD_CACHE is useful when the number of rows in the table is small or you expect to access a high percentage of the table values.
   NO_CACHE reads values from the lookup table for every row without caching values. Select DEMAND_LOAD_CACHE when the number of rows in the table is large and you expect to frequently access a low percentage of table values or when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.
7. To provide more resources to execute the lookup_ext function, select *Run as a separate process*. This option creates a separate child data flow process for the lookup_ext function when the software executes the data flow.
8. Define one or more conditions. For each, add a lookup table column name (select from the drop-down list or drag from the *Parameter* pane), select the appropriate operator, and enter an expression by typing, dragging, pasting, or using the Smart Editor (click the icon in the right column).
   In the example, the condition is `ID_DEPT = Employees.ID_DEPT`.
9. Define the output. For each output column:

   a) Add a lookup table column name.

   b) Optionally change the default value from NULL.

   c) Specify the *Output column name* by typing, dragging, pasting, or using the Smart Editor (click the icon in the right column).

   In the example, the output column is ID_DEPT_NAME.
10. If multiple matches are possible, specify the ordering and set a return policy (default is *MAX*) to select one match. To order the output, enter the column name(s) in the *Order by* list.

Example

The following example illustrates how to use the lookup table ID_lookup.txt to retrieve department names for employees.

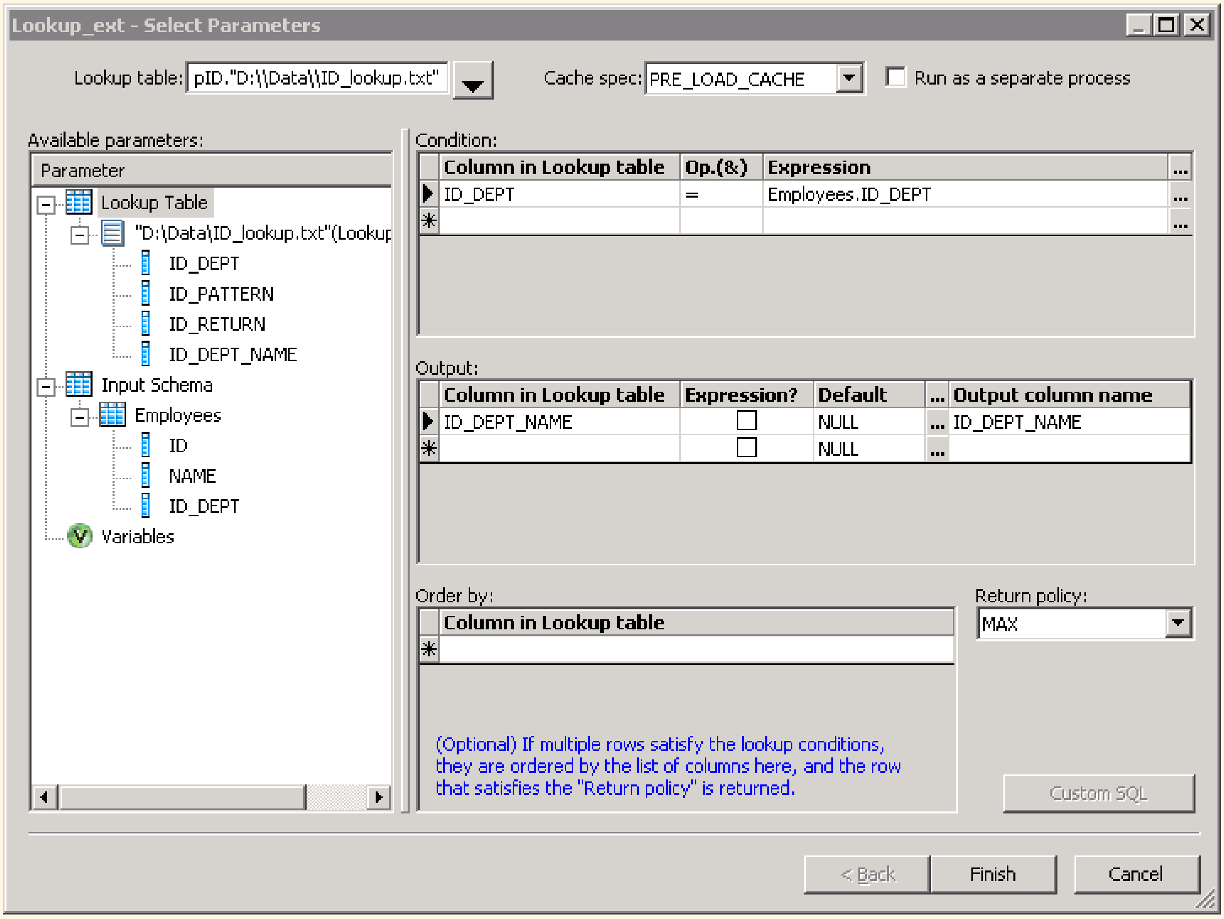
The Employees table is as follows:

ID	NAME	ID_DEPT
SSN111111111	Employee1	10
SSN222222222	Employee2	10
TAXID333333333	Employee3	20

The lookup table ID_lookup.txt is as follows:

ID_DEPT	ID_PATTERN	ID_RETURN	ID_DEPT_NAME
10	ms(SSN*)	=substr(ID_Pattern,4,20)	Payroll
20	ms(TAXID*)	=substr(ID_Pattern,6,30)	Accounting

The lookup_ext editor would be configured as follows.



Lookup table: pID."D:\\Data\\ID_lookup.txt"

Cache spec: PRE_LOAD_CACHE

☐ Run as a separate process

Parameter
- Lookup Table
  - "D:\Data\ID_lookup.txt"(Lookup
    - ID_DEPT
    - ID_PATTERN
    - ID_RETURN
    - ID_DEPT_NAME
- Input Schema
  - Employees
    - ID
    - NAME
    - ID_DEPT
- Variables

Condition:

Column in Lookup table	Op.(&)	Expression
ID_DEPT	=	Employees.ID_DEPT

Output:

Column in Lookup table	Expression?	Default	Output column name
ID_DEPT_NAME	☐	NULL	ID_DEPT_NAME
	☐	NULL	

Order by:

Column in Lookup table

(Optional) If multiple rows satisfy the lookup conditions, they are ordered by the list of columns here, and the row that satisfies the "Return policy" is returned.

Return policy: MAX

Related Information

Example: Defining a complex lookup_ext function [page 297]

2.7.5.3 Example: Defining a complex lookup_ext function

This procedure describes the process for defining a complex lookup_ext function using a new function call. The associated example uses the same lookup and input tables as in the *Example: Defining a simple lookup_ext function* [page 295] This example illustrates how to extract and normalize employee ID numbers.

For details on all the available options for the lookup_ext function, see the *Reference Guide*.

1. In a data flow, open the Query editor.
2. From the *Schema in* pane, drag the ID column to the *Schema out* pane. Do the same for the Name column.
3. In the *Schema out* pane, right-click the Name column and click *New Function Call*. Click *Insert Below*.
4. Select the *Function category Lookup Functions* and the *Function name lookup_ext* and click *Next*.

5. In the *Lookup_ext - Select Parameters* window, select a lookup table:
   In the example, the lookup table is in the file format ID_lookup.txt that is in D:\Data.
6. Define one or more conditions.
   In the example, the condition is `ID_PATTERN ~ Employees.ID`.
7. Define the output. For each output column:
   a) Add a lookup table column name.
   b) If you want the software to interpret the column in the lookup table as an expression and return the calculated value, select the *Expression* check box.
   c) Optionally change the default value from NULL.
   d) Specify the *Output column name*(s) by typing, dragging, pasting, or using the Smart Editor (click the icon in the right column).

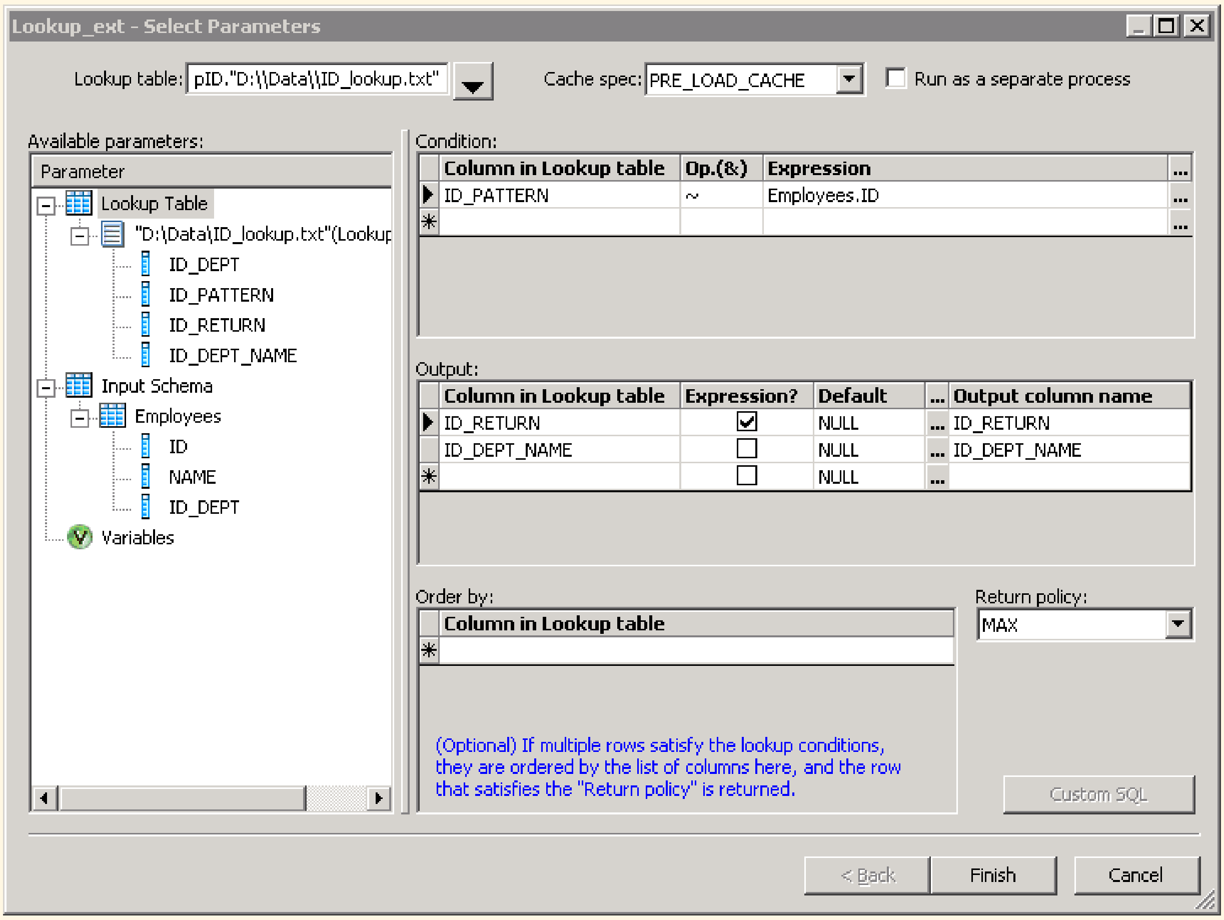
   In the example, the output columns are ID_RETURN and ID_DEPT_NAME.

Example

Extract and normalize employee ID numbers

In this example, you want to extract and normalize employee Social Security numbers and tax identification numbers that have different prefixes. You want to remove the prefixes, thereby normalizing the numbers. You also want to identify the department from where the number came. The data flow has one source table Employees, a query configured with lookup_ext, and a target table.

Configure the lookup_ext editor as in the following graphic.


Lookup_ext - Select Parameters

Lookup table: pID."D:\\Data\\ID_lookup.txt"

Cache spec: PRE_LOAD_CACHE

☐ Run as a separate process

Available parameters:

Condition:

Column in Lookup table	Op.(&)	Expression
ID_PATTERN	~	Employees.ID

Output:

Column in Lookup table	Expression?	Default	Output column name
ID_RETURN	☑	NULL	ID_RETURN
ID_DEPT_NAME	☐	NULL	ID_DEPT_NAME
	☐	NULL	

Order by:

Column in Lookup table

Return policy: MAX

(Optional) If multiple rows satisfy the lookup conditions, they are ordered by the list of columns here, and the row that satisfies the "Return policy" is returned.

The lookup condition is `ID_PATTERN ~ Employees.ID`.

The software reads each row of the source table Employees, then checks the lookup table ID_lookup.txt for all rows that satisfy the lookup condition.

The operator ~ means that the software will apply a pattern comparison to Employees.ID. When it encounters a pattern in ID_lookup.ID_PATTERN that matches Employees.ID, the software applies the expression in ID_lookup.ID_RETURN. In this example, Employee1 and Employee2 both have IDs that match the pattern ms(SSN*) in the lookup table. the software then applies the expression `=substr(ID_PATTERN,4,20)` to the data, which extracts from the matched string (Employees.ID) a substring of up to 20 characters starting from the 4th position. The results for Employee1 and Employee2 are 111111111 and 222222222, respectively.

For the output of the ID_RETURN lookup column, the software evaluates ID_RETURN as an expression because the *Expression* box is checked. In the lookup table, the column ID_RETURN contains the expression `=substr(ID_PATTERN,4,20)`. ID_PATTERN in this expression refers to the lookup table column ID_PATTERN. When the lookup condition `ID_PATTERN ~ Employees.ID` is true, the software evaluates the expression. Here the software substitutes the placeholder ID_PATTERN with the actual Employees.ID value.

The output also includes the ID_DEPT_NAME column, which the software returns as a literal value (because the Expression box is not checked). The resulting target table is as follows:

ID	NAME	ID_RETURN	ID_DEPT_NAME
SSN111111111	Employee1	111111111	Payroll

ID	NAME	ID_RETURN	ID_DEPT_NAME
SSN222222222	Employee2	222222222	Payroll
TAXID333333333	Employee3	333333333	Accounting

Related Information

Reference Guide: Functions and Procedures, lookup_ext [page 1605]
Data Flows, Accessing the lookup_ext editor [page 294]
Data Flows, Example: Defining a simple lookup_ext function [page 295]
Reference Guide: Functions and Procedures, match_simple [page 1630]

2.7.6 Data flow execution

A data flow is a declarative specification from which the software determines the correct data to process. For example in data flows placed in batch jobs, the transaction order is to extract, transform, then load data into a target. Data flows are similar to SQL statements. The specification declares the desired output.

The software executes a data flow each time the data flow occurs in a job. However, you can specify that a batch job execute a particular data flow only one time. In that case, the software only executes the first occurrence of the data flow; the software skips subsequent occurrences in the job.

You might use this feature when developing complex batch jobs with multiple paths, such as jobs with try/catch blocks or conditionals, and you want to ensure that the software only executes a particular data flow one time.

Related Information

Creating and defining data flows [page 284]

2.7.6.1 Push down operations to the database server

From the information in the data flow specification, the software produces output while optimizing performance. For example, for SQL sources and targets, the software creates database-specific SQL statements based on a job's data flow diagrams. To optimize performance, the software pushes down as many transform operations as possible to the source or target database and combines as many operations as possible into one request to the database. For example, the software tries to push down joins and function evaluations. By pushing down operations to the database, the software reduces the number of rows and operations that the engine must process.

Data flow design influences the number of operations that the software can push to the source or target database. Before running a job, you can examine the SQL that the software generates and alter your design to produce the most efficient results.

You can use the Data_Transfer transform to push down resource-intensive operations anywhere within a data flow to the database. Resource-intensive operations include joins, GROUP BY, ORDER BY, and DISTINCT.

Related Information

Performance Optimization Guide: Maximizing push-down operations [page 2116]
Reference Guide: Data_Transfer [page 1071]

2.7.6.2 Distributed data flow execution

The software provides capabilities to distribute CPU-intensive and memory-intensive data processing work (such as join, grouping, table comparison and lookups) across multiple processes and computers. This work distribution provides the following potential benefits:

- Better memory management by taking advantage of more CPU resources and physical memory
- Better job performance and scalability by using concurrent sub data flow execution to take advantage of grid computing

You can create sub data flows so that the software does not need to process the entire data flow in memory at one time. You can also distribute the sub data flows to different job servers within a server group to use additional memory and CPU resources.

Use the following features to split a data flow into multiple sub data flows:

- *Run as a separate process* option on resource-intensive operations that include the following:
  - Hierarchy_Flattening transform
  - Associate transform
  - Country ID transform
  - Global Address Cleanse transform
  - Global Suggestion Lists transform
  - Match Transform
  - United States Regulatory Address Cleanse transform
  - User-Defined transform
  - Query operations that are CPU-intensive and memory-intensive:
    - Join
    - GROUP BY
    - ORDER BY
    - DISTINCT
  - Table_Comparison transform
  - Lookup_ext function
  - Count_distinct function

    - Search_replace function

  If you select the Run as a separate process option for multiple operations in a data flow, the software splits the data flow into smaller sub data flows that use separate resources (memory and computer) from each other. When you specify multiple Run as a separate process options, the sub data flow processes run in parallel.
- Data_Transfer transform
  With this transform, the software does not need to process the entire data flow on the Job Server computer. Instead, the Data_Transfer transform can push down the processing of a resource-intensive operation to the database server. This transform splits the data flow into two sub data flows and transfers the data to a table in the database server to enable the software to push down the operation.

Related Information

Performance Optimization Guide: Splitting a data flow into sub data flows [page 2153]
Performance Optimization Guide: Data_Transfer transform for push-down operations [page 2122]

2.7.6.3 Load balancing

You can distribute the execution of a job or a part of a job across multiple Job Servers within a Server Group to better balance resource-intensive operations. You can specify the following values on the *Distribution level* option when you execute a job:

- Job level—A job can execute on an available Job Server.
- Data flow level—Each data flow within a job can execute on an available Job Server.
- Sub data flow level—An resource-intensive operation (such as a sort, table comparison, or table lookup) within a data flow can execute on an available Job Server.

Related Information

Performance Optimization Guide: Using grid computing to distribute data flows execution [page 2160]

2.7.6.4 Caches

The software provides the option to cache data in memory to improve operations such as the following in your data flows.

- Joins—Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join.
- Table comparisons—Because a comparison table must be read for each row of a source, you might want to cache the comparison table.

- Lookups—Because a lookup table might exist on a remote database, you might want to cache it in memory to reduce access times.

The software provides the following types of caches that your data flow can use for all of the operations it contains:

- In-memory—Use in-memory cache when your data flow processes a small amount of data that fits in memory.
- Pageable cache—Use a pageable cache when your data flow processes a very large amount of data that does not fit in memory.

If you split your data flow into sub data flows that each run on a different Job Server, each sub data flow can use its own cache type.

Related Information

Performance Optimization Guide: Using Caches [page 2129]

2.7.7 Audit Data Flow overview

You can audit objects within a data flow to collect run time audit statistics. You can perform the following tasks with this auditing feature:

- Collect audit statistics about data read into a job, processed by various transforms, and loaded into targets.
- Define rules about the audit statistics to determine if the correct data is processed.
- Generate notification of audit failures.
- Query the audit statistics that persist in the repository.

Related Information

Using Auditing [page 452]

2.8 Transforms

Transforms operate on data sets by manipulating input sets and producing one or more output sets. By contrast, functions operate on single values in specific columns in a data set.

Many built-in transforms are available from the object library on the *Transforms* tab.

The transforms that you can use depend on the software package that you have purchased. (If a transform belongs to a package that you have not purchased, it is disabled and cannot be used in a job.)

Transforms are grouped into the following categories:

- Data Integrator: Transforms that allow you to extract, transform, and load data. These transform help ensure data integrity and maximize developer productivity for loading and updating data warehouse environment.
- Data Quality: Transforms that help you improve the quality of your data. These transforms can parse, standardize, correct, enrich, match and consolidate your customer and operational information assets.
- Platform: Transforms that are needed for general data movement operations. These transforms allow you to generate, map and merge rows from two or more sources, create SQL query operations (expressions, lookups, joins, and filters), perform conditional splitting, and so on.
- Text Data Processing: Transforms that help you extract specific information from your text. They can parse large volumes of text, which allows you to identify and extract entities and facts, such as customers, products, locations, and financial information relevant to your organization.

Transform Category	Transform	Description
Data Integrator	Data_Transfer	Allows a data flow to split its processing into two sub data flows and push down resource-consuming operations to the database server.
	Date_Generation	Generates a column filled with date values based on the start and end dates and increment that you provide.
	Effective_Date	Generates an additional "effective to" column based on the primary key's "effective date."
	Hierarchy_Flattening	Flattens hierarchical data into relational tables so that it can participate in a star schema. Hierarchy flattening can be both vertical and horizontal.
	History_Preserving	Converts rows flagged as UPDATE to UPDATE plus INSERT, so that the original values are preserved in the target. You specify in which column to look for updated data.
	Key_Generation	Generates new keys for source data, starting from a value based on existing keys in the table you specify.
	Map_CDC_Operation	Sorts input data, maps output data, and resolves before- and after-images for UPDATE rows. While commonly used to support Oracle changed-data capture, this transform supports any data stream if its input requirements are met.
	Pivot (Columns to Rows)	Rotates the values in specified columns to rows. (Also see Reverse Pivot.)
	Reverse Pivot (Rows to Columns)	Rotates the values in specified rows to columns.
	Table_Comparison	Compares two data sets and produces the difference between them as a data set with rows flagged as INSERT and UPDATE.
	XML_Pipeline	Processes large XML inputs in small batches.
Data Quality	Associate	Combine the results of two or more Match transforms or two or more Associate transforms, or any

Transform Category	Transform	Description
		combination of the two, to find matches across match sets.
	Country ID	Parses input data and then identifies the country of destination for each record.
	Data Cleanse	Identifies and parses name, title, and firm data, phone numbers, Social Security numbers, dates, and e-mail addresses. It can assign gender, add prenames, generate Match standards, and convert input sources to a standard format. It can also parse and manipulate various forms of international data, as well as operational and product data.
	DSF2 Walk Sequencer	Adds delivery sequence information to your data, which you can use with presorting software to qualify for walk-sequence discounts.
	Geocoder	Uses geographic coordinates, addresses, and point-of-interest (POI) data to append address, latitude and longitude, census, and other information to your records.
	Global Address Cleanse	Identifies, parses, validates, and corrects global address data, such as primary number, primary name, primary type, directional, secondary identifier, and secondary number.
	Global Suggestion Lists	Completes and populates addresses with minimal data, and it can offer suggestions for possible matches.
	Match	Identifies matching records based on your business rules. Also performs candidate selection, unique ID, best record, and other operations.
	USA Regulatory Address Cleanse	Identifies, parses, validates, and corrects USA address data according to the U.S. Coding Accuracy Support System (CASS).
	User-Defined	Does just about anything that you can write Python code to do. You can use the User-Defined transform to create new records and data sets, or populate a field with a specific value, just to name a few possibilities.
Platform	Case	Simplifies branch logic in data flows by consolidating case or decision making logic in one transform. Paths are defined in an expression table.
	Map_Operation	Modifies data based on current operation codes and mapping expressions. The operation codes can then

Transform Category	Transform	Description
		be converted between data manipulation operations.
	Merge	Unifies rows from two or more sources into a single target.
	Query	Retrieves a data set that satisfies conditions that you specify. A query transform is similar to a SQL SELECT statement.
	Row_Generation	Generates a column filled with integer values starting at zero and incrementing by one to the end value you specify.
	SQL	Performs the indicated SQL query operation.
	Validation	Ensures that the data at any stage in the data flow meets your criteria. You can filter out or replace data that fails your criteria.
Text Data Processing	Entity_Extraction	Extracts information (entities and facts) from any text, HTML, XML , or binary format content such as PDF.

Related Information

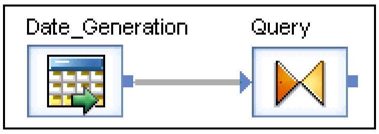
Reference Guide: Transforms [page 1067]

2.8.1 To add a transform to a data flow

You can use the Designer to add transforms to data flows.

1. Open a data flow object.
2. Open the object library if it is not already open and click the *Transforms* tab.
3. Select the transform or transform configuration that you want to add to the data flow.
4. Drag the transform or transform configuration icon into the data flow workspace. If you selected a transform that has available transform configurations, a drop-down menu prompts you to select a transform configuration.
5. Draw the data flow connections.

   To connect a source to a transform, click the square on the right edge of the source and drag the cursor to the arrow on the left edge of the transform.



Continue connecting inputs and outputs as required for the transform.

  ○ The input for the transform might be the output from another transform or the output from a source; or, the transform may not require source data.
  ○ You can connect the output of the transform to the input of another transform or target.

6. Double-click the name of the transform.

   This opens the transform editor, which lets you complete the definition of the transform.
7. Enter option values.

   To specify a data column as a transform option, enter the column name as it appears in the input schema or drag the column name from the input schema into the option box.

Related Information

To add a Query transform to a data flow [page 310]
To add a Data Quality transform to a data flow [page 312]
To add a text data processing transform to a data flow [page 324]

2.8.2 Transform editors

After adding a transform to a data flow, you configure it using the transform's editor. Transform editor layouts vary.

The most commonly used transform is the Query transform, which has two panes:

- An input schema area and/or output schema area
- A options area (or parameters area) that lets you to set all the values the transform requires

Data Quality transforms, such as Match and Data Cleanse, use a transform editor that lets you set options and map input and output fields.

The Entity Extraction transform editor lets you set extraction options and map input and output fields.

Related Information

Query Editor [page 311]
Data Quality transform editors [page 314]
Entity Extraction transform editor [page 325]

2.8.3 Transform configurations

A transform configuration is a transform with preconfigured best practice input fields, best practice output fields, and options that can be used in multiple data flows. These are useful if you repeatedly use a transform with specific options and input and output fields.

Some transforms, such as Data Quality transforms, have read-only transform configurations that are provided when Data Services is installed. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one. You cannot perform export or multi-user operations on read-only transform configurations.

In the Transform Configuration Editor window, you set up the default options, best practice input fields, and best practice output fields for your transform configuration. After you place an instance of the transform configuration in a data flow, you can override these preset defaults.

If you edit a transform configuration, that change is inherited by every instance of the transform configuration used in data flows, unless a user has explicitly overridden the same option value in an instance.

Related Information

To create a transform configuration [page 308]
To add a user-defined field [page 309]

2.8.3.1 To create a transform configuration

1. In the *Transforms* tab of the *Local Object Library,* right-click a transform and select *New* to create a new transform configuration, or right-click an existing transform configuration and select *Replicate*.

   If New or Replicate is not available from the menu, then the selected transform type cannot have transform configurations.

   The *Transform Configuration Editor* window opens.
2. In *Transform Configuration Name*, enter the name of the transform configuration.
3. In the *Options* tab, set the option values to determine how the transform will process your data. The available options depend on the type of transform that you are creating a configuration for.

   For the Associate, Match, and User-Defined transforms, options are not editable in the Options tab. You must set the options in the Associate Editor, Match Editor, or User-Defined Editor, which are accessed by clicking the *Edit Options* button.

   If you change an option value from its default value, a green triangle appears next to the option name to indicate that you made an override.
4. To designate an option as "best practice," select the *Best Practice* checkbox next to the option's value. Designating an option as best practice indicates to other users who use the transform configuration which options are typically set for this type of transform.

   Use the filter to display all options or just those options that are designated as best practice options.
5. Click the *Verify* button to check whether the selected option values are valid.

If there are any errors, they are displayed at the bottom of the window.

6. In the *Input Best Practices* tab, select the input fields that you want to designate as the best practice input fields for the transform configuration.

   The transform configurations provided with Data Services do not specify best practice input fields, so that it doesn't appear that one input schema is preferred over other input schemas. For example, you may map the fields in your data flow that contain address data whether the address data resides in discrete fields, multiline fields, or a combination of discrete and multiline fields.

   These input fields will be the only fields displayed when the Best Practice filter is selected in the Input tab of the transform editor when the transform configuration is used within a data flow.
7. For Associate, Match, and User-Defined transform configurations, you can create user-defined input fields. Click the *Create* button and enter the name of the input field.
8. In the *Output Best Practices* tab, select the output fields that you want to designate as the best practice output fields for the transform configuration.
   These output fields will be the only fields displayed when the Best Practice filter is selected in the Output tab of the transform editor when the transform configuration is used within a data flow.
9. Click *OK* to save the transform configuration.
   The transform configuration is displayed in the *Local Object Library* under the base transform of the same type.

You can now use the transform configuration in data flows.

Related Information

Reference Guide: Transforms, Transform configurations [page 1123]

2.8.3.2 To add a user-defined field

For some transforms, such as the Associate, Match, and User-Defined transforms, you can create user-defined input fields rather than fields that are recognized by the transform. These transforms use user-defined fields because they do not have a predefined set of input fields.

You can add a user-defined field either to a single instance of a transform in a data flow or to a transform configuration so that it can be used in all instances.

In the User-Defined transform, you can also add user-defined output fields.

1. In the *Transforms* tab of the *Local Object Library,* right-click an existing Associate, Match, or User-Defined transform configuration and select *Edit*.
   The *Transform Configuration Editor* window opens.
2. In the *Input Best Practices* tab, click the *Create* button and enter the name of the input field.
3. Click *OK* to save the transform configuration.

When you create a user-defined field in the transform configuration, it is displayed as an available field in each instance of the transform used in a data flow. You can also create user-defined fields within each transform instance.

Related Information

Data Quality transform editors [page 314]

2.8.4 The Query transform

The Query transform is by far the most commonly used transform, so this section provides an overview.

The Query transform can perform the following operations:

- Choose (filter) the data to extract from sources
- Join data from multiple sources
- Map columns from input to output schemas
- Perform transformations and functions on the data
- Perform data nesting and unnesting
- Add new columns, nested schemas, and function results to the output schema
- Assign primary keys to output columns

Related Information

Nested Data [page 342]

Reference Guide: Transforms [page 1067]

2.8.4.1 To add a Query transform to a data flow

Because it is so commonly used, the Query transform icon is included in the tool palette, providing an easier way to add a Query transform.

1. Click the Query icon in the tool palette.
2. Click anywhere in a data flow workspace.
3. Connect the Query to inputs and outputs.

> Note
>
> - The inputs for a Query can include the output from another transform or the output from a source.
> - The outputs from a Query can include input to another transform or input to a target.
> - You can change the content type for the columns in your data by selecting a different type from the output content type list.
> - If you connect a target table to a Query with an empty output schema, the software automatically fills the Query's output schema with the columns from the target table, without mappings.

2.8.4.2 Query Editor

The Query Editor is a graphical interface for performing query operations. It contains the following areas: *input schema* area (upper left), *output schema* area (upper right), and a *parameters* area (lower tabbed area). The ⓘ icon indicates that the tab contains user-defined entries or that there is at least one join pair (FROM tab only).

The input and output schema areas can contain: Columns, Nested schemas, and Functions (output only).

The *Schema In* and *Schema Out* lists display the currently selected schema in each area. The currently selected output schema is called the current schema and determines the following items:

- The output elements that can be modified (added, mapped, or deleted)
- The scope of the *Select* through *Order by* tabs in the parameters area

The current schema is highlighted while all other (non-current) output schemas are gray.

2.8.4.2.1 To change the current output schema

You can change the current output schema in the following ways:

- Select a schema from the Output list so that it is highlighted.
- Right-click a schema, column, or function in the Output Schema area and select *Make Current*.
- Double-click one of the non-current (grayed-out) elements in the Output Schema area.

2.8.4.2.2 To modify the output schema contents

You can modify the output schema in several ways:

- Drag and drop (or copy and paste) columns or nested schemas from the input schema area to the output schema area to create simple mappings.
- Use right-click menu options on output elements to:
  - Add new output columns and schemas.
  - Use function calls to generate new output columns.
  - Assign or reverse primary key settings on output columns. Primary key columns are flagged by a key icon.
  - Unnest or re-nest schemas.
- Use the *Mapping* tab to provide complex column mappings. Drag and drop input schemas and columns into the output schema to enable the editor. Use the function wizard and the smart editor to build expressions. When the text editor is enabled, you can access these features using the buttons above the editor.
- Use the *Select* through *Order By* tabs to provide additional parameters for the current schema (similar to SQL SELECT statement clauses). You can drag and drop schemas and columns into these areas.

Tab name	Description
Select	Specifies whether to output only distinct rows (discarding any identical duplicate rows).

Tab name	Description
From	Lists all input schemas. Allows you to specify join pairs and join conditions as well as enter join rank and cache for each input schema. The resulting SQL FROM clause is displayed.

Where	Specifies conditions that determine which rows are output.
	Enter the conditions in SQL syntax, like a WHERE clause in a SQL SELECT statement. For example:
	`<TABLE1.EMPNO> = <TABLE2.EMPNO> AND`
	`<TABLE1.EMPNO> > 1000 OR`
	`<TABLE2.EMPNO> < 9000`
	Use the *Functions*, *Domains*, and *smart editor* buttons for help building expressions.

Group By	Specifies how the output rows are grouped (if required).
Order By	Specifies how the output rows are sequenced (if required).

- Use the *Find* tab to locate input and output elements containing a specific word or term.

2.8.5 Data Quality transforms

Data Quality transforms are a set of transforms that help you improve the quality of your data. The transforms can parse, standardize, correct, and append information to your customer and operational data.

Data Quality transforms include the following transforms:

- Associate
- Country ID
- Data Cleanse
- DSF2 Walk Sequencer
- Global Address Cleanse
- Global Suggestion Lists
- Match
- USA Regulatory Address Cleanse
- User-Defined

Related Information

Reference Guide: Transforms [page 1067]

2.8.5.1 To add a Data Quality transform to a data flow

Data Quality transforms cannot be directly connected to an upstream transform that contains or generates nested tables. This is common in real-time data flows, especially those that perform matching. To connect these transforms, you must insert either a Query transform or an XML Pipeline transform between the transform with the nested table and the Data Quality transform.

1. Open a data flow object.
2. Open the object library if it is not already open.
3. Go to the *Transforms* tab.
4. Expand the Data Quality transform folder and select the transform or transform configuration that you want to add to the data flow.
5. Drag the transform or transform configuration icon into the data flow workspace. If you selected a transform that has available transform configurations, a drop-down menu prompts you to select a transform configuration.
6. Draw the data flow connections.

   To connect a source or a transform to another transform, click the square on the right edge of the source or upstream transform and drag the cursor to the arrow on the left edge of the Data Quality transform.

   - The input for the transform might be the output from another transform or the output from a source; or, the transform may not require source data.
   - You can connect the output of the transform to the input of another transform or target.
7. Double-click the name of the transform.

   This opens the transform editor, which lets you complete the definition of the transform.
8. In the input schema, select the input fields that you want to map and drag them to the appropriate field in the *Input* tab.
   This maps the input field to a field name that is recognized by the transform so that the transform knows how to process it correctly. For example, an input field that is named "Organization" would be mapped to the Firm field. When content types are defined for the input, these columns are automatically mapped to the appropriate input fields. You can change the content type for the columns in your data by selecting a different type from the output content type list.
9. For the Associate, Match, and User-Defined transforms, you can add user-defined fields to the *Input* tab. You can do this in two ways:

   - Click the first empty row at the bottom of the table and press F2 on your keyboard. Enter the name of the field. Select the appropriate input field from the drop-down box to map the field.
   - Drag the appropriate input field to the first empty row at the bottom of the table.

   To rename the user-defined field, click the name, press F2 on your keyboard, and enter the new name.
10. In the *Options* tab, select the appropriate option values to determine how the transform will process your data.

    - Make sure that you map input fields before you set option values, because in some transforms, the available options and option values depend on the mapped input fields.
    - For the Associate, Match, and User-Defined transforms, options are not editable in the Options tab. You must set the options in the Associate Editor, Match Editor, and User-Defined Editor. You can access these editors either by clicking the Edit Options button in the Options tab or by right-clicking the transform in the data flow.

    If you change an option value from its default value, a green triangle appears next to the option name to indicate that you made an override.
11. In the *Output* tab, double-click the fields that you want to output from the transform. Data Quality transforms can generate fields in addition to the input fields that the transform processes, so you can output many fields.

Make sure that you set options before you map output fields.

The selected fields appear in the output schema. The output schema of this transform becomes the input schema of the next transform in the data flow.

12. If you want to pass data through the transform without processing it, drag fields directly from the input schema to the output schema.
13. To rename or resize an output field, double-click the output field and edit the properties in the *Column Properties* window.

Related Information

Reference Guide: Data Quality Fields [page 1130]

Data Quality transform editors [page 314]

2.8.5.2 Data Quality transform editors

The Data Quality editors, graphical interfaces for setting input and output fields and options, contain the following areas: input schema area (upper left), output schema area (upper right), and the parameters area (lower tabbed area).

The parameters area contains three tabs: Input, Options, and Output. Generally, it is considered best practice to complete the tabs in this order, because the parameters available in a tab may depend on parameters selected in the previous tab.

Input schema area

The input schema area displays the input fields that are output from the upstream transform in the data flow.

Output schema area

The output schema area displays the fields that the transform outputs, and which become the input fields for the downstream transform in the data flow.

Input tab

The Input tab displays the available field names that are recognized by the transform. You map these fields to input fields in the input schema area. Mapping input fields to field names that the transform recognizes tells the transform how to process that field.

Options tab

The Options tab contain business rules that determine how the transform processes your data. Each transform has a different set of available options. If you change an option value from its default value, a green triangle appears next to the option name to indicate that you made an override.

In the Associate, Match, and User-Defined transforms, you cannot edit the options directly in the Options tab. Instead you must use the Associate, Match, and User-Defined editors, which you can access from the Edit Options button.

Output tab

The Output tab displays the field names that can be output by the transform. Data Quality transforms can generate fields in addition to the input fields that that transform processes, so that you can output many fields. These mapped output fields are displayed in the output schema area.

Filter and sort

The Input, Options, and Output tabs each contain filters that determine which fields are displayed in the tabs.

Filter	Description
Best Practice	Displays the fields or options that have been designated as a best practice for this type of transform. However, these are merely suggestions; they may not meet your needs for processing or outputting your data. The transform configurations provided with the software do not specify best practice input fields.
In Use	Displays the fields that have been mapped to an input field or output field.
All	Displays all available fields.

The Output tab has additional filter and sort capabilities that you access by clicking the column headers. You can filter each column of data to display one or more values, and also sort the fields in ascending or descending order. Icons in the column header indicate whether the column has a filter or sort applied to it. Because you can filter and sort on multiple columns, they are applied from left to right. The filter and sort menu is not available if there is only one item type in the column.

Embedded help

The embedded help is the place to look when you need more information about Data Services transforms and options. The topic changes to help you with the context you're currently in. When you select a new transform or a new option group, the topic updates to reflect that selection.

You can also navigate to other topics by using hyperlinks within the open topic.

Associate, Match, and User-Defined transform editors

To view option information for the Associate, Match, and User-Defined transforms, you will need to open their respective editors by selecting the transform in the data flow and then choosing *Tools* > `<transform>` *Editor*

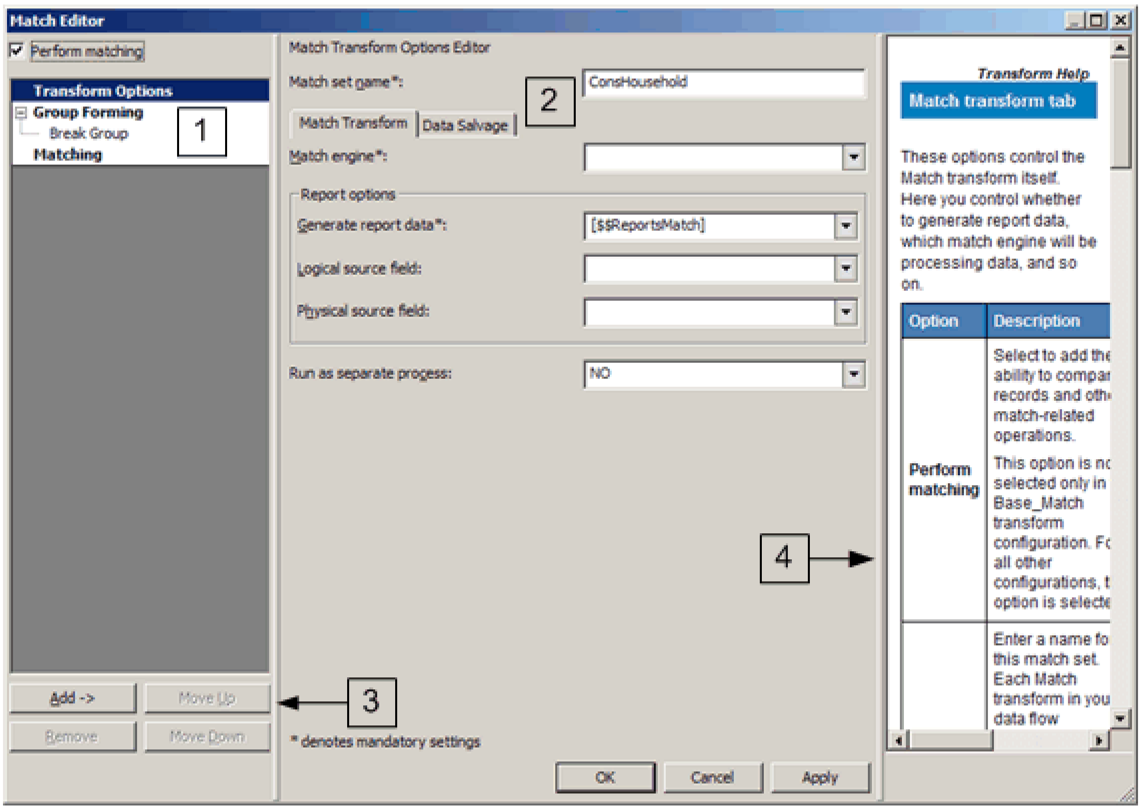
> **Note**
>
> You cannot access the Match editor options while you are reviewing data flows created in Information Steward Data Cleansing Advisor. Match transform options cannot be edited; therefore, controls to access the Match editor are inactive.

Related Information

Associate, Match, and User-Defined transform editors [page 316]

2.8.5.2.1 Associate, Match, and User-Defined transform editors

The Associate, Match, and User-Defined transforms each have their own editor in which you can add option groups and edit options. The editors for these three transforms look and act similarly, and in some cases even share the same option groups.



☑ Perform matching

Match Transform Options Editor

Match set name*: ConsHousehold

Match engine*:

Generate report data*: [$$ReportsMatch]

Logical source field:

Physical source field:

Run as separate process: NO

* denotes mandatory settings

Transform Help

Match transform tab

These options control the Match transform itself. Here you control whether to generate report data, which match engine will be processing data, and so on.

Option	Description
Perform matching	Select to add the ability to compar records and oth match-related operations. This option is nc selected only in Base_Match transform configuration. Fc all other configurations, t option is selecte
	Enter a name fo this match set. Each Match transform in you data flow

The editor window is divided into four areas:

1. *Option Explorer* — In this area, you select the option groups, or operations, that are available for the transform. To display an option group that is hidden, right-click the option group it belongs to and select the name of the option group from the menu.
2. *Option Editor* — In this area, you specify the value of the option.
3. *Buttons* — Use these to add, remove and order option groups.
4. *Embedded help* — The embedded help displays additional information about using the current editor screen.

Related Information

Reference Guide: Transforms, Associate [page 1132]
Reference Guide: Transforms, Match [page 1263]
Reference Guide: Transforms, User-Defined [page 1371]

2.8.5.2.2 Ordered options editor

Some transforms allow you to choose and specify the order of multiple values for a single option. One example is the parser sequence option of the Data Cleanse transform.

To configure an ordered option:

1. Click the *Add* and *Remove* buttons to move option values between the Available and Selected values lists.

   ℹ Note

   Remove all values. To clear the Selected values list and move all option values to the Available values list, click *Remove All*.

2. Select a value in the Available values list, and click the up and down arrow buttons to change the position of the value in the list.
3. Click *OK* to save your changes to the option configuration. The values are listed in the Designer and separated by pipe characters.

2.8.6 Text Data Processing transforms

Text Data Processing transforms help you extract specific information from your text. They can parse large volumes of text, identifying and extracting entities and facts, such as customers, products, locations, and financial information relevant to your organization. The following sections provide an overview of this functionality and the Entity Extraction transform.

2.8.6.1 Text Data Processing overview

Text Data Processing can automatically identify the input text language and select language-specific dictionaries and rules for analysis to extract entities, including people, dates, places, organizations and so on, in the languages. It also looks for patterns, activities, events, and relationships among entities and enables their extraction. Extracting such information from text tells you what the text is about — this information can be used within applications for information management, data integration, and data quality; business intelligence; query, analytics and reporting; search, navigation, document and content management; among other usage scenarios.

Text Data Processing goes beyond conventional character matching tools for information retrieval, which can only seek exact matches for specific strings. It understands semantics of words. In addition to known entity matching, it performs a complementary function of new entity discovery. To customize entity extraction, the software enables you to specify your own list of entities in a custom dictionary. These dictionaries enable you to store entities and manage name variations. Known entity names can be standardized using a dictionary.

Text Data Processing automates extraction of key information from text sources to reduce manual review and tagging. This in turn can reduce cost towards understanding important insights hidden in text. Access to relevant information from unstructured text can help streamline operations and reduce unnecessary costs.

In Data Services, text data processing refers to a set of transforms that extracts information from unstructured data and creates structured data that can be used by various business intelligence tools.

2.8.6.2 Entity Extraction transform overview

Text data processing is accomplished in the software using the following transform:

- Entity Extraction - Extracts entities and facts from unstructured text.

Extraction involves processing and analyzing text identifying languages, finding entities of interest, assigning them to the appropriate type, and presenting this metadata in a standard format. By using dictionaries and rules, you can customize your extraction output to include entities defined in them. Extraction applications are as diverse as your information needs. Some examples of information that can be extracted using this transform include:

- Co-occurrence and associations of brand names, company names, people, supplies, and more.
- Competitive and market intelligence such as competitors' activities, merger and acquisition events, press releases, contact information, and so on.
- A person's associations, activities, or role in a particular event.
- Customer claim information, defect reports, or patient information such as adverse drug effects.
- Various alphanumeric patterns such as ID numbers, contract dates, profits, and so on.

2.8.6.2.1 Entities and facts overview

Entities denote names of people, places, and things that can be extracted. *Entities* are defined as a pairing of a name and its type. *Type* indicates the main category of an entity.

Here are some examples of entities:

- Paris is an entity with name "Paris" and type *LOCALITY*.
- Mr. Joe Smith is an entity with name "Mr. Joe Smith" and type *PERSON*.

Entities can have subtypes. A subtype indicates further classification of an entity; it is a hierarchical specification of an entity type that enables the distinction between different semantic varieties of the same entity type. A subtype can be described as a sub-category of an entity.

Here are some examples of entities and subtypes:

- Boeing 747 is an entity of type *VEHICLE* with a subtype *AIR*.
- Mercedes-Benz SL500 is an entity of type *VEHICLE* with a subtype *LAND*.
- SAP is an entity of type *ORGANIZATION* with a subtype *COMMERCIAL*.

Facts denote a pattern that creates an expression to extract information such as sentiments, events, or relationships. Facts are extracted using custom extraction rules. Fact is an umbrella term covering extractions of more complex patterns including one or more entities, a relationship between one or more entities, or some sort of predicate about an entity. Facts provide context of how different entities are connected in the text. Entities by themselves only show that they are present in a document, but facts provide information on how these entities are related. Fact types identify the category of a fact; for example, sentiments and requests. A *subfact* is a key piece of information embedded within a fact. A subfact type can be described as a category associated with the subfact.

Here are some examples of facts and fact types:

- SAP acquired Business Objects in a friendly takeover. This is an event of type merger and acquisition (*M&A*).
- Mr. Joe Smith is very upset with his airline bookings. This is a fact of type *SENTIMENT*.

How extraction works

The extraction process uses its inherent knowledge of the semantics of words and the linguistic context in which these words occur to find entities and facts. It creates specific patterns to extract entities and facts based on system rules. You can add entries in a dictionary as well as write custom rules to customize extraction output. The following sample text and sample output shows how unstructured content can be transformed into structured information for further processing and analysis.

Example

Sample text and extraction information

"Mr. Jones is very upset with Green Insurance Corp. The offer for his totaled vehicle is too low. He states that Green offered him $1250.00 but his car is worth anywhere from $2500 to $4500. Mr. Jones would like Green's comprehensive coverage to be in line with other competitors."

This sample text when processed with the extraction transform, configured with the sentiment and request custom rules would identify and group the information in a logical way (identifying entities, subtypes, facts, fact types, subfacts, and subfact types) that can be further processed.

The following tables show partial results with information tagged as entities, entity types, subtypes, facts, fact types, subfacts, and subfact types from the sample text:

Entities	Entity type	Subtype
Mr. Jones	*PERSON*	
Green Insurance	*ORGANIZATION*	*COMMERCIAL*
Green	*PROP_MISC*	
$1250.00	*CURRENCY*	

Facts	Fact type	Subfact	Subfact type
Mr. Jones is very upset with Green Insurance Corp.	*SENTIMENT*	*very upset*	*StrongNegativeSentiment*
Jones would like that Green's comprehensive coverage to be in line with other competitors.	*REQUEST*		

2.8.6.2.2 Dictionary overview

An extraction dictionary is a user-defined repository of entities. It is an easy-to-use customization tool that specifies a list of entities that the Entity Extraction transform should always extract while processing text. The information is classified under the *standard form* and the *variant* of an entity. A standard form may have one or more variants embedded under it; variants are other commonly known names of an entity. For example, *United*

Parcel Service of America is the standard form for that company, and *United Parcel Service* and *UPS* are both variants for the same company.

While each standard form must have a type, variants can optionally have their own type; for example, while *United Parcel Service of America* is associated with a standard form type *ORGANIZATION*, you might define a variant type *ABBREV* to include abbreviations. A dictionary structure can help standardize references to an entity.

For more information, see “Using Dictionaries” in the *Text Data Processing Extraction Customization Guide*.

2.8.6.2.3 Rule overview

An extraction rule defines custom patterns to extract entities, relationships, events, and other larger extractions that are together referred to as facts. You write custom extraction rules to perform extraction that is customized to your specific needs.

For more information, see “Using Extraction Rules” in the *Text Data Processing Extraction Customization Guide*.

2.8.6.3 Using the Entity Extraction transform

The Entity Extraction transform can extract information from any text, HTML, XML, or certain binary-format (such as PDF) content and generate structured output. You can use the output in several ways based on your work flow. You can use it as an input to another transform or write to multiple output sources such as a database table or a flat file. The output is generated in `UTF-16` encoding. The following list provides some scenarios on when to use the transform alone or in combination with other Data Services transforms.

- Searching for specific information and relationships from a large amount of text related to a broad domain. For example, a company is interested in analyzing customer feedback received in free form text after a new product launch.
- Linking structured information from unstructured text together with existing structured information to make new connections. For example, a law enforcement department is trying to make connections between various crimes and people involved using their own database and information available in various reports in text format.
- Analyzing and reporting on product quality issues such as excessive repairs and returns for certain products. For example, you may have structured information about products, parts, customers, and suppliers in a database, while important information pertaining to problems may be in notes: fields of maintenance records, repair logs, product escalations, and support center logs. To identify the issues, you need to make connections between various forms of data.

2.8.6.4 Differences between text data processing and data cleanse transforms

The Entity Extraction transform provides functionality similar to the Data Cleanse transform in certain cases, especially with respect to customization capabilities. This section describes the differences between the two and which transform to use to meet your goals. The Text Data Processing Entity Extraction transform is for making sense of unstructured content and the Data Cleanse transform is for standardizing and cleansing structured data. The following table describes some of the main differences. In many cases, using a combination of Entity Extraction and Data Cleanse transforms will generate the data that is best suited for your business intelligence analyses and reports.

Criteria	Text Data Processing	Data Cleanse
Input type	Unstructured text that requires linguistic parsing to generate relevant information.	Structured data represented as fields in records.
Input size	More than 5KB of text.	Less than 5KB of text.
Input scope	Normally broad domain with many variations.	Specific data domain with limited variations.
Matching task	Content discovery, noise reduction, pattern matching, and relationship between different entities.	Dictionary lookup, pattern matching.
Potential usage	Identifies potentially meaningful information from unstructured content and extracts it into a format that can be stored in a repository.	Ensures quality of data for matching and storing into a repository such as Meta Data Management.
Output	Creates annotations about the source text in the form of entities, entity types, facts, and their offset, length, and so on. Input is not altered.	Creates parsed and standardized fields. Input is altered if desired.

2.8.6.5 Using multiple transforms

You can include multiple transforms in the same data flow to perform various analytics on unstructured information.

For example, to extract names and addresses embedded in some text and validate the information before running analytics on the extracted information, you could:

- Use the Entity Extraction transform to process text containing names and addresses and extract different entities.
- Pass the extraction output to the Case transform to identify which rows represent names and which rows represent addresses
- Use the Data Cleanse transform to standardize the extracted names and use the Global Address Cleanse transform to validate and correct the extracted address data.

Note

To generate the correct data, include the `standard_form` and `type` fields in the Entity Extraction transform output schema; map the `type` field in the Case transform based on the entity type such as `PERSON`, `ADDRESS1`, etc. Next, map any `PERSON` entities from the Case transform to the Data Cleanse transform and map any `ADDRESS1` entities to the Global Address Cleanse transform.

2.8.6.6 Examples for using the Entity Extraction transform

This section describes some examples for employing the Entity Extraction transform.

The scenario is that a human resources department wants to analyze résumés received in a variety of formats. The formats include:

- A text file as an attachment to an email
- A text résumé pasted into a field on the company's Web site
- Updates to résumé content that the department wants to process in real time

Example

Text file email attachment

The human resources department frequently receives résumés as attachments to emails from candidates. They store these attachments in a separate directory on a server.

To analyze and process data from these text files:

1. Configure an *Unstructured text* file format that points to the directory of résumés.
2. Build a data flow with the unstructured text file format as the source, an Entity Extraction transform, and a target.
3. Configure the transform to process and analyze the text.

Example

Text résumé pasted into a field on a Web site

The human resources department's online job application form includes a field into which applicants can paste their résumés. This field is captured in a database table column.

To analyze and process data from the database:

1. Configure a connection to the database via a datastore.
2. Build a data flow with the database table as the source, an Entity Extraction transform, and a target.
3. Configure the transform to process and analyze the text.

Example

Updated content to be processed in real time

Suppose the human resources department is seeking a particular qualification in an applicant. When the applicant updates her résumé in the company's Web-based form with the desired qualification, the HR manager wants to be immediately notified. Use a real-time job to enable this functionality.

To analyze and process the data in real time:

1. Add a real-time job including begin and end markers and a data flow. Connect the objects.
2. Build the data flow with a message source, an Entity Extraction transform, and a message target.
3. Configure the transform to process and analyze the text.

Related Information

Unstructured file formats [page 280]
Database datastores [page 210]
Real-time Jobs [page 372]

2.8.6.7 To add a text data processing transform to a data flow

1. Open a data flow object.
2. Open the local object library if it is not already open.
3. Go to the *Transforms* tab.
4. Expand the *Text Data Processing* transform folder and select the transform or transform configuration that you want to add to the data flow.
5. Drag the transform or transform configuration icon into the data flow workspace. If you selected a transform that has available transform configurations, a drop-down menu prompts you to select a transform configuration.
6. Draw the data flow connections.

   To connect a source or a transform to another transform, click the square on the right edge of the source or upstream transform and drag the cursor to the arrow on the left edge of the text data processing transform.

   - The input for the transform might be the output from another transform or the output from a source.
   - You can connect the output of the transform to the input of another transform or target.
7. Double-click the name of the transform.

   This opens the transform editor, which lets you complete the definition of the transform.
8. In the input schema, select the input field that you want to map and drag it to the appropriate field in the *Input* tab.
   This maps the input field to a field name that is recognized by the transform so that the transform knows how to process it correctly. For example,

   - an input field that is named *Content* would be mapped to the *TEXT* input field.
   - an input field that can uniquely identify the content would be mapped to the *TEXT_ID* input field.
9. In the *Options* tab, select the appropriate option values to determine how the transform will process your data.

   Make sure that you map input fields before you set option values.

   If you change an option value from its default value, a green triangle appears next to the option name to indicate that you made an override.
10. In the *Output* tab, double-click the fields that you want to output from the transform. The transforms can generate fields in addition to the input fields that the transform processes, so you can output many fields.

    Make sure that you set options before you map output fields.

    The selected fields appear in the output schema. The output schema of this transform becomes the input schema of the next transform in the data flow.

11. If you want to pass data through the transform without processing it, drag fields directly from the input schema to the output schema.
12. To rename or resize an output field, double-click the output field and edit the properties in the *Column Properties* window.

Related Information

Using Extraction Options [page 325]
Reference Guide: Entity Extraction transform, Input fields [page 1507]
Reference Guide: Entity Extraction transform, Output fields [page 1508]
Reference Guide: Entity Extraction transform, Extraction options [page 1502]

2.8.6.8 Entity Extraction transform editor

The *Entity Extraction* transform options specify various parameters to process content using the transform. Filtering options, under different extraction options, enable you to limit the entities and facts extracted to specific entities from a dictionary, the system files, entities/facts from rules, or a combination of them.

Extraction options are divided into the following categories:

- *Common*
  This option is set to specify that the Entity Extraction transform is to be run as a separate process.
- *Languages*
  Use this option to specify the language for the extraction process.
  - *Language*—The default is 'Auto', directing the transform to attempt to identify the language. You may select another language from a list.
  - *Default Language*—You may select the language that should be assumed if the transform was not able to identify the language.
  - *Filter by Entity Types* is optional and you may select it when you select the language to limit your extraction output.
- *Processing Options*
  Use these options to specify parameters to be used when processing the content.
- *Dictionaries*
  Use this option to specify different dictionaries to be used for processing the content. To use the *Entity Types* filtering option, you must specify the *Dictionary File*.

  > Note
  >
  > Text Data Processing includes the dictionary schema file `extraction-dictionary.xsd`. By default, this file is installed in the `LINK_DIR/bin` folder, where `LINK_DIR` is your Data Services installation directory. Refer to this schema to create your own dictionary files.

- *Rules*
  Use this option to specify different rule files to be used for processing the content. To use the *Rule Names* filtering option, you must specify the *Rule File*.

If you do not specify any filtering options, the extraction output will contain all entities extracted using entity types defined in the selected language, dictionary file(s), and rule name(s) in the selected rule file(s).

> **Note**
>
> Selecting a dictionary file or a rule file in the extraction process is optional. The extraction output will include the entities from them if they are specified.

For more information, see “Using Dictionaries” in the *Text Data Processing Extraction Customization Guide*.

Related Information

Importing XML Schemas [page 346]

Reference Guide: Entity Extraction transform, Extraction options [page 1502]

2.8.6.9 Using filtering options

The filtering options under different extraction options control the output generated by the Entity Extraction transform. Using these options, you can limit the entities extracted to specific entities from a dictionary, the system files, entities/facts from rules, or a combination of them. For example, you are processing customer feedback fields for an automobile company and are interested in looking at the comments related to one specific model. Using the filtering options, you can control your output to extract data only related to that model.

Filtering options are divided into three categories:

- The *Filter By Entity Types* option under the *Languages* option group—Use this option to limit extraction output to include only selected entities for this language.
- The *Filter By Entity Types* option under the *Dictionary* option group—Use this option to limit extraction output to include only entities defined in a dictionary.
- The *Filter By Rules Names* option under the *Rules* option group—Use this option to limit extraction output to include only entities and facts returned by the specific rules.

The following table describes information contained in the extraction output based on the combination of these options:

Languages	Dictionaries	Rules	Extraction output content	Note
Entity Types	Entity Types	Rule Names		
Yes	No	No	Entities (extracted using the entity types) selected in the filter.	
No	Yes	No	Entities (extracted using the entity types) defined in the selected language and entity types selected from the dictionaries filter.	If multiple dictionaries are specified that contain the same entity type but this entity type is selected as a filter for only one of them, entities of this

Languages	Dictionaries	Rules	Extraction output content	Note
				type will also be returned from the other dictionary.
Yes	Yes	No	Entities (extracted using the entity types) defined in the filters for the selected language and any specified dictionaries.	
No	No	Yes	Entities (extracted using the entity types) defined in the selected language and any rule names selected in the filter from any specified rule files.	If multiple rule files are specified that contain the same rule name but it is only selected as a filter for one of them, entities and facts of this type will also be returned from the other rule file.
No	Yes	Yes	Entities (extracted using entity types) defined in the selected language, entity types selected from the dictionaries filter, and any rule names selected in the filter from any specified rule files.	
Yes	No	Yes	Entities (extracted using entity types) defined in the filters for the selected language and any rule names selected in the filter from any specified rule files.	
Yes	Yes	Yes	Entities (extracted using entity types) defined in the filters for the selected language, entity types selected from the dictionaries filter, and any rule names selected in the filter from any specified rule files.	The extraction process filters the output using the union of the extracted entities or facts for the selected language, the dictionaries, and the rule files.

If you change your selection for the language, dictionaries, or rules, any filtering associated with that option will only be cleared by clicking the *Filter by...* option. You must select new filtering choices based on the changed selection.

i Note

If you are using multiple dictionaries (or rules) and have set filtering options for some of the selected dictionaries (or rules), the extraction process combines the dictionaries internally, and output is filtered using the union of the entity types selected for each dictionary and rule names selected for each rule file. The output will identify the source as a dictionary (or rule) file and not the individual name of a dictionary (or rule) file.

i Note

If you select the *Dictionary Only* option under the *Processing Options* group, with a valid dictionary file, the entity types defined for the language are not included in the extraction output, but any extracted rule file entities and facts are included.

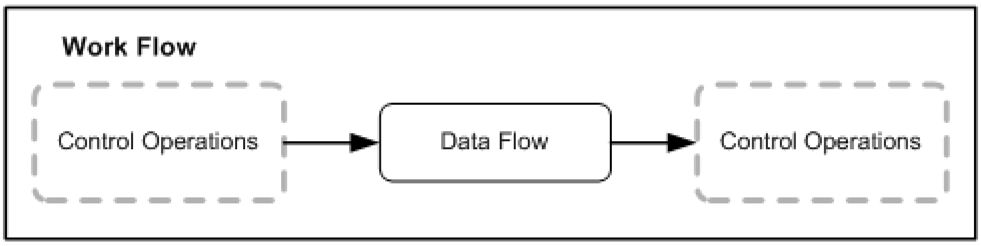
Related Information

Entity Extraction transform editor [page 325]

2.9 Work Flows

2.9.1 What is a work flow?

A work flow defines the decision-making process for executing data flows. For example, elements in a work flow can determine the path of execution based on a value set by a previous job or can indicate an alternative path if something goes wrong in the primary path. Ultimately, the purpose of a work flow is to prepare for executing data flows and to set the state of the system after the data flows are complete.



Jobs are special work flows. Jobs are special because you can execute them. Almost all of the features documented for work flows also apply to jobs, with one exception: jobs do not have parameters.

Related Information

Projects [page 205]

2.9.2 Steps in a work flow

Work flow steps take the form of icons that you place in the work space to create a work flow diagram. The following objects can be elements in work flows:

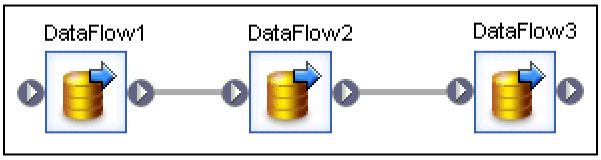
- Work flows
- Data flows
- Scripts
- Conditionals
- While loops
- Try/catch blocks

Work flows can call other work flows, and you can nest calls to any depth. A work flow can also call itself.

The connections you make between the icons in the workspace determine the order in which work flows execute, unless the jobs containing those work flows execute in parallel.

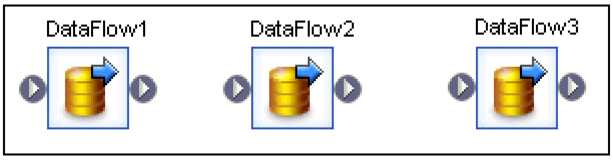
2.9.3 Order of execution in work flows

Steps in a work flow execute in a left-to-right sequence indicated by the lines connecting the steps. Here is the diagram for a work flow that calls three data flows:

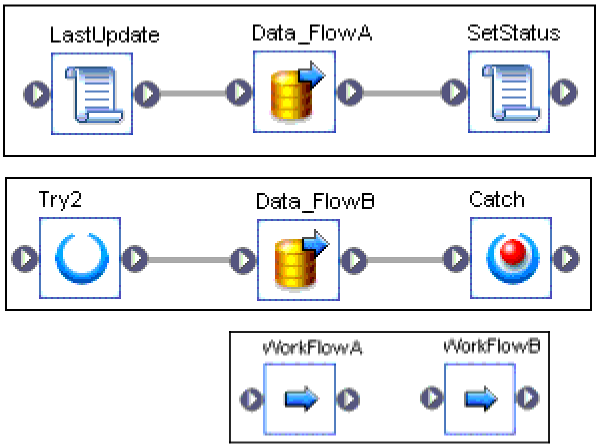


Note that Data_Flow1 has no connection from the left but is connected on the right to the left edge of Data_Flow2 and that Data_Flow2 is connected to Data_Flow3. There is a single thread of control connecting all three steps. Execution begins with Data_Flow1 and continues through the three data flows.

Connect steps in a work flow when there is a dependency between the steps. If there is no dependency, the steps need not be connected. In that case, the software can execute the independent steps in the work flow as separate processes. In the following work flow, the software executes data flows 1 through 3 in parallel:



To execute more complex work flows in parallel, define each sequence as a separate work flow, then call each of the work flows from another work flow as in the following example:



You can specify that a job execute a particular work flow or data flow only one time. In that case, the software only executes the first occurrence of the work flow or data flow; the software skips subsequent occurrences in the job. You might use this feature when developing complex jobs with multiple paths, such as jobs with try/catch blocks or conditionals, and you want to ensure that the software only executes a particular work flow or data flow one time.

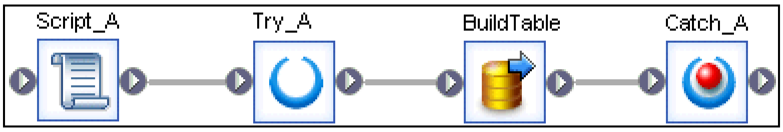
2.9.4 Example of a work flow

Suppose you want to update a fact table. You define a data flow in which the actual data transformation takes place. However, before you move data from the source, you want to determine when the fact table was last updated so that you only extract rows that have been added or changed since that date.

You need to write a script to determine when the last update was made. You can then pass this date to the data flow as a parameter.

In addition, you want to check that the data connections required to build the fact table are active when data is read from them. To do this in the software, you define a try/catch block. If the connections are not active, the catch runs a script you wrote, which automatically sends mail notifying an administrator of the problem.

Scripts and error detection cannot execute in the data flow. Rather, they are steps of a decision-making process that influences the data flow. This decision-making process is defined as a work flow, which looks like the following:



The software executes these steps in the order that you connect them.

2.9.5 Creating work flows

You can create work flows using one of two methods:

- Object library
- Tool palette

After creating a work flow, you can specify that a job only execute the work flow one time, as a single process, or as a continous process even if the work flow appears in the job multiple times.

2.9.5.1 To create a new work flow using the object library

1. Open the object library.
2. Go to the *Work Flows* tab.
3. Right-click and choose *New*.

4. Drag the work flow into the diagram.
5. Add the data flows, work flows, conditionals, try/catch blocks, and scripts that you need.

2.9.5.2 To create a new work flow using the tool palette

1. Select the work flow icon in the tool palette.
2. Click where you want to place the work flow in the diagram.

If more than one instance of a work flow appears in a job, you can improve execution performance by running the work flow only one time.

2.9.5.3 To specify that a job executes the work flow one time

When you specify that a work flow should only execute once, a job will never re-execute that work flow after the work flow completes successfully, except if the work flow is contained in a work flow that is a recovery unit that re-executes and has not completed successfully elsewhere outside the recovery unit.

It is recommended that you not mark a work flow as *Execute only once* if the work flow or a parent work flow is a recovery unit.

1. Right click on the work flow and select *Properties*.

   The Properties window opens for the work flow.
2. Select *Regular* from the *Execution type* dropdown list.
3. Select the *Execute only once* check box.
4. Click *OK*.

Related Information

Reference Guide: Work flow [page 1002]

2.9.5.4 What is a single work flow?

A single work flow runs all of its child data flows in one operating system process. If the data flows are designed to be run in parallel then they are run in different threads instead of different processes. The advantage of single process is that it is possible to share resources such as database connections across multiple data flows.

> Note
>
> Single work flows have the following limitations:
>
> - A single work flow cannot call or contain a continuous work flow.

- A single work flow cannot use sub data flows. Therefore, the Data Transfer transform and "Run as a separate process" options are invalid. The software will generate a runtime validation error.
- A single work flow can be only executed by a continuous work flow. A single work flow cannot call another single work flow.

2.9.5.5 To specify that a job executes as a single work flow

1. Right-click a work flow and select *Properties*
2. Select *Single* from the Execution type dropdown list.
3. Click *OK*.

2.9.5.6 What is a continuous work flow?

A continuous work flow runs all data flows in a loop but keeps them in the memory for the next iteration. This eliminates the need to repeat some of the common steps of execution (for example, connecting to the repository, parsing/optimizing/compiling ATL, opening database connections.

Note

Continuous work flows have the following limitations:

- A continuous work flow cannot call or contain another continuous work flow. If a continuous work flow calls another continous work flow which never terminates, the work flow can never restart the child processes.
- A continuous work flow cannot use sub data flows. Therefore, the Data Transfer transform and "Run as a separate process" options are invalid. The software will generate a runtime validation error.
- A regular or single work flow cannot call or contain a continuous work flow.
- A real-time job cannot call a continuous work flow.
- The platform transform XML_Map cannot be used in a continuous work flow.
- The Data Integrator transforms, Data_Transfer, Table_Comparison, and XML_Pipeline cannot be used in a continuous work flow.

2.9.5.7 To specify that a job executes a continuous work flow

1. Right-click a work flow and select *Properties*
2. Select *Continuous* from the Execution type dropdown list.
3. Access the *Continuous Options* tab.
4. Specify when you want the work flow to release resources:
   - To release resources after a number of runs, select *Number of runs* and enter the number of runs. The default is 100.
   - To release resources after a number of hours, select the *After* checkbox, select *Number of hours*, and enter the number of hours.

- To release resources after a number of days, select the *After* checkbox, select *Number of days*, and enter the number of days.
- To release resources when the result of a function is not equal to zero, select the *After* checkbox, select *Result of the function is not equal to zero*, and enter the function you want to use.

5. To stop the work flow when the result of a custom function is equal to zero, select *When result of the function is equal to zero*, and enter the custom function you want to use.
6. Click *OK.*

2.9.6 Conditionals

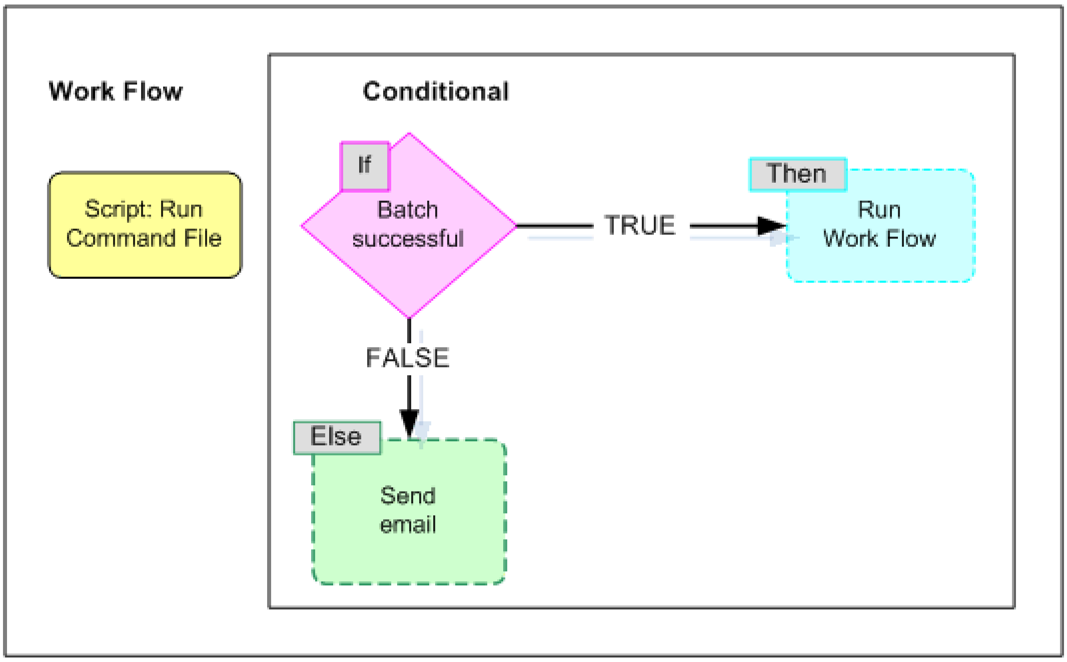
Conditionals are single-use objects used to implement if/then/else logic in a work flow. Conditionals and their components (if expressions, then and else diagrams) are included in the scope of the parent control flow's variables and parameters.

To define a conditional, you specify a condition and two logical branches:

Conditional branch	Description
If	A Boolean expression that evaluates to TRUE or FALSE. You can use functions, variables, and standard operators to construct the expression.
Then	Work flow elements to execute if the *If* expression evaluates to TRUE.
Else	(Optional) Work flow elements to execute if the *If* expression evaluates to FALSE.

Define the *Then* and *Else* branches inside the definition of the conditional.

A conditional can fit in a work flow. Suppose you use a Windows command file to transfer data from a legacy system into the software. You write a script in a work flow to run the command file and return a success flag. You then define a conditional that reads the success flag to determine if the data is available for the rest of the work flow.



To implement this conditional in the software, you define two work flows—one for each branch of the conditional. If the elements in each branch are simple, you can define them in the conditional editor itself.

Both the *Then* and *Else* branches of the conditional can contain any object that you can have in a work flow including other work flows, nested conditionals, try/catch blocks, and so on.

2.9.6.1 To define a conditional

1. Define the work flows that are called by the *Then* and *Else* branches of the conditional.

   It is recommended that you define, test, and save each work flow as a separate object rather than constructing these work flows inside the conditional editor.
2. Open the work flow in which you want to place the conditional.
3. Click the icon for a conditional in the tool palette.
4. Click the location where you want to place the conditional in the diagram.

   The conditional appears in the diagram.
5. Click the name of the conditional to open the conditional editor.
6. Click *if*.
7. Enter the Boolean expression that controls the conditional.

   Continue building your expression. You might want to use the function wizard or smart editor.
8. After you complete the expression, click *OK*.
9. Add your predefined work flow to the *Then* box.

   To add an existing work flow, open the object library to the Work Flows tab, select the desired work flow, then drag it into the Then box.

10. (Optional) Add your predefined work flow to the *Else* box.

    If the If expression evaluates to FALSE and the Else box is blank, the software exits the conditional and continues with the work flow.
11. After you complete the conditional, choose *DebugValidate*.

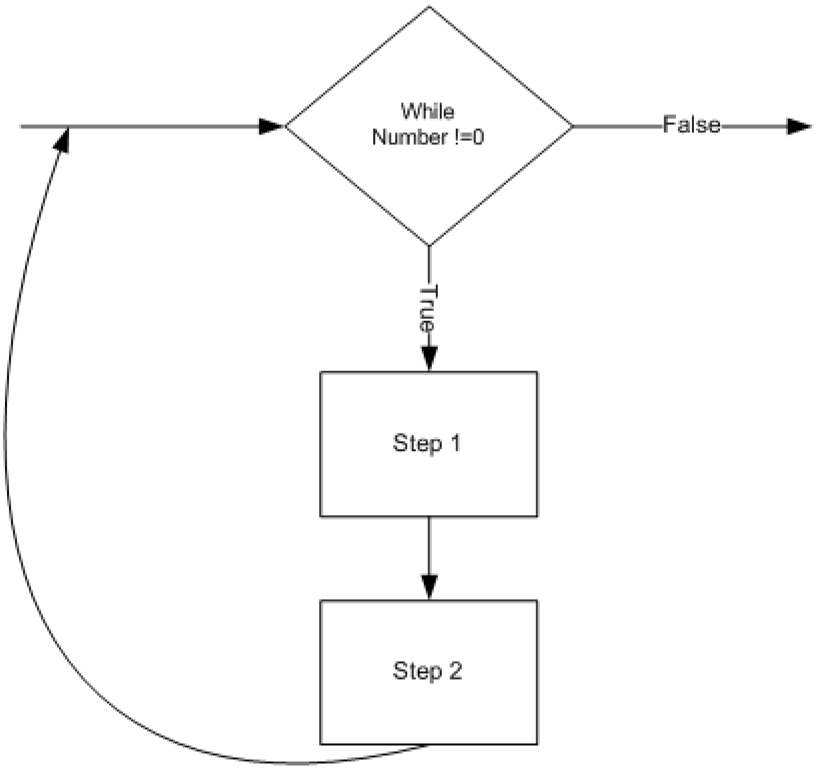
    The software tests your conditional for syntax errors and displays any errors encountered.
12. The conditional is now defined. Click the *Back* button to return to the work flow that calls the conditional.

2.9.7 While loops

Use a while loop to repeat a sequence of steps in a work flow as long as a condition is true.

2.9.7.1 Design considerations

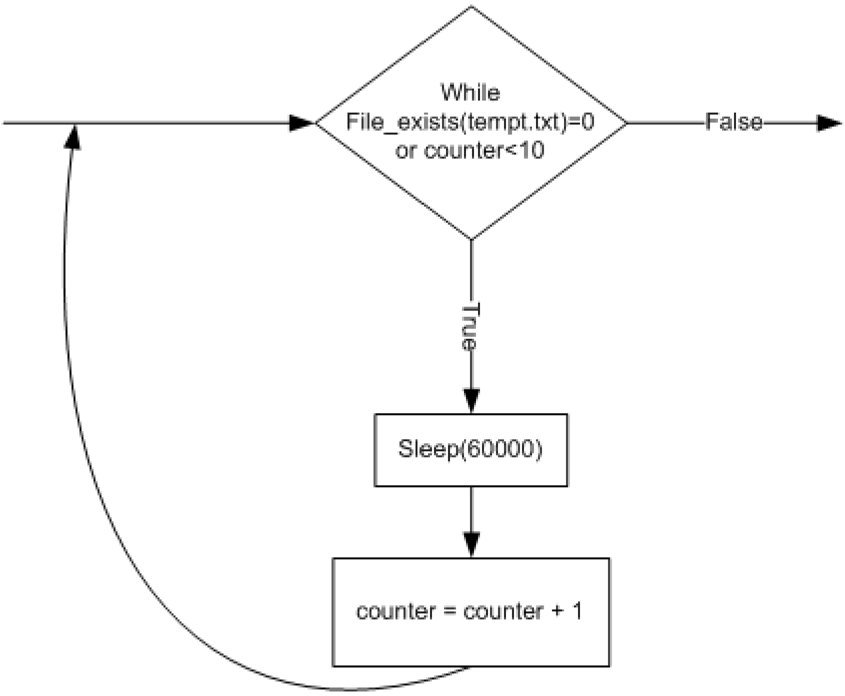
The while loop is a single-use object that you can use in a work flow. The while loop repeats a sequence of steps as long as a condition is true.



Typically, the steps done during the while loop result in a change in the condition so that the condition is eventually no longer satisfied and the work flow exits from the while loop. If the condition does not change, the while loop will not end.

For example, you might want a work flow to wait until the system writes a particular file. You can use a while loop to check for the existence of the file using the `file_exists` function. As long as the file does not exist, you can have the work flow go into sleep mode for a particular length of time, say one minute, before checking again.

Because the system might never write the file, you must add another check to the loop, such as a counter, to ensure that the while loop eventually exits. In other words, change the while loop to check for the existence of the file and the value of the counter. As long as the file does not exist and the counter is less than a particular value, repeat the while loop. In each iteration of the loop, put the work flow in sleep mode and then increment the counter.



2.9.7.2 Defining a while loop

You can define a while loop in any work flow.

2.9.7.2.1 To define a while loop

1. Open the work flow where you want to place the while loop.
2. Click the while loop icon on the tool palette.
3. Click the location where you want to place the while loop in the workspace diagram.

   The while loop appears in the diagram.
4. Click the while loop to open the while loop editor.
5. In the *While* box at the top of the editor, enter the condition that must apply to initiate and repeat the steps in the while loop.

   Alternatively, click to open the expression editor, which gives you more space to enter an expression and access to the function wizard. Click OK after you enter an expression in the editor.

6. Add the steps you want completed during the while loop to the workspace in the while loop editor.

   You can add any objects valid in a work flow including scripts, work flows, and data flows. Connect these objects to represent the order that you want the steps completed.

   > **Note**
   >
   > Although you can include the parent work flow in the while loop, recursive calls can create an infinite loop.

7. After defining the steps in the while loop, choose *Debug* > *Validate*.

   The software tests your definition for syntax errors and displays any errors encountered.
8. Close the while loop editor to return to the calling work flow.

2.9.7.3 Using a while loop with View Data

When using View Data, a job stops when the software has retrieved the specified number of rows for all scannable objects.

Depending on the design of your job, the software might not complete all iterations of a while loop if you run a job in view data mode:

- If the while loop contains scannable objects and there are no scannable objects outside the while loop (for example, if the while loop is the last object in a job), then the job will complete after the scannable objects in the while loop are satisfied, possibly after the first iteration of the while loop.
- If there are scannable objects after the while loop, the while loop will complete normally. Scanned objects in the while loop will show results from the last iteration.
- If there are no scannable objects following the while loop but there are scannable objects completed in parallel to the while loop, the job will complete as soon as all scannable objects are satisfied. The while loop might complete any number of iterations.

2.9.8 Try/catch blocks

A try/catch block is a combination of one try object and one or more catch objects that allow you to specify alternative work flows if errors occur while the software is executing a job. Try/catch blocks:

- "Catch" groups of exceptions "thrown" by the software, the DBMS, or the operating system.
- Apply solutions that you provide for the exceptions groups or for specific errors within a group.
- Continue execution.

Try and catch objects are single-use objects.

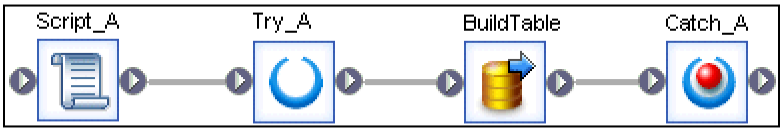
Here's the general method to implement exception handling:

1. Insert a try object before the steps for which you are handling errors.
2. Insert a catch object in the work flow after the steps.
3. In the catch object, do the following:
   - ○ Select one or more groups of errors that you want to catch.

- Define the actions that a thrown exception executes. The actions can be a single script object, a data flow, a work flow, or a combination of these objects.
- Optional. Use catch functions inside the catch block to identify details of the error.

If an exception is thrown during the execution of a try/catch block and if no catch object is looking for that exception, then the exception is handled by normal error logic.

The following work flow shows a try/catch block surrounding a data flow:



In this case, if the data flow BuildTable causes any system-generated exceptions specified in the catch Catch_A, then the actions defined in Catch_A execute.

The action initiated by the catch object can be simple or complex. Here are some examples of possible exception actions:

- Send the error message to an online reporting database or to your support group.
- Rerun a failed work flow or data flow.
- Run a scaled-down version of a failed work flow or data flow.

Related Information

Defining a Try/Catch block [page 338]

Categories of available exceptions [page 340]

Example: Catching details of an error [page 340]

Reference Guide: Objects, Catch [page 846]

2.9.8.1 Defining a try/catch block

To define a try/catch block:

1. Open the work flow that will include the try/catch block.
2. Click the try icon in the tool palette.
3. Click the location where you want to place the try in the diagram.

   The try icon appears in the diagram.

   > Note
   >
   > There is no editor for a try; the try merely initiates the try/catch block.

4. Click the catch icon in the tool palette.

5. Click the location where you want to place the catch object in the work space.

   The catch object appears in the work space.
6. Connect the try and catch objects to the objects they enclose.
7. Click the name of the catch object to open the catch editor.
8. Select one or more groups from the list of *Exceptions*.

   To select all exception groups, click the check box at the top.
9. Define the actions to take for each exception group and add the actions to the catch work flow box. The actions can be an individual script, a data flow, a work flow, or any combination of these objects.
   a) It is recommended that you define, test, and save the actions as a separate object rather than constructing them inside the catch editor.
   b) If you want to define actions for specific errors, use the following catch functions in a script that the work flow executes:
      - error_context()
      - error_message()
      - error_number()
      - error_timestamp()
   c) To add an existing work flow to the catch work flow box, open the object library to the Work Flows tab, select the desired work flow, and drag it into the box.
10. After you have completed the catch, choose *Validation* > *Validate* > *All Objects in View*.

    The software tests your definition for syntax errors and displays any errors encountered.
11. Click the *Back* button to return to the work flow that calls the catch.
12. If you want to catch multiple exception groups and assign different actions to each exception group, repeat steps 4 through 11 for each catch in the work flow.

    > Note
    >
    > In a sequence of catch blocks, if one catch block catches an exception, the subsequent catch blocks will not be executed. For example, if your work flow has the following sequence and Catch1 catches an exception, then Catch2 and CatchAll will not execute.
    >
    > ```
    > Try > DataFlow1 > Catch1 > Catch2 > CatchAll
    > ```

If any error in the exception group listed in the catch occurs during the execution of this try/catch block, the software executes the catch work flow.

Related Information

Categories of available exceptions [page 340]
Example: Catching details of an error [page 340]
Reference Guide: Objects, Catch [page 846]

2.9.8.2 Categories of available exceptions

Categories of available exceptions include:

- Execution errors (1001)
- Database access errors (1002)
- Database connection errors (1003)
- Flat file processing errors (1004)
- File access errors (1005)
- Repository access errors (1006)
- SAP system errors (1007)
- System resource exception (1008)
- SAP BW execution errors (1009)
- XML processing errors (1010)
- COBOL copybook errors (1011)
- Excel book errors (1012)
- Data Quality transform errors (1013)

2.9.8.3 Example: Catching details of an error

This example illustrates how to use the error functions in a catch script. Suppose you want to catch database access errors and send the error details to your support group.

1. In the catch editor, select the exception group that you want to catch. In this example, select the checkbox in front of *Database access errors (1002)*.
2. In the work flow area of the catch editor, create a script object with the following script:

```
mail_to('support@my.com',
  'Data Service error number' || error_number(),
  'Error message: ' || error_message(),20,20);
print('DBMS Error: ' || error_message());
```

3. This sample catch script includes the mail_to function to do the following:
   - Specify the email address of your support group.
   - Send the error number that the error_number() function returns for the exception caught.
   - Send the error message that the error_message() function returns for the exception caught.
4. The sample catch script includes a print command to print the error message for the database error.

Related Information

Reference Guide: Objects, Catch error functions [page 846]

Reference Guide: Objects, Catch scripts [page 846]

2.9.9 Scripts

Scripts are single-use objects used to call functions and assign values to variables in a work flow.

For example, you can use the SQL function in a script to determine the most recent update time for a table and then assign that value to a variable. You can then assign the variable to a parameter that passes into a data flow and identifies the rows to extract from a source.

A script can contain the following statements:

- Function calls
- If statements
- While statements
- Assignment statements
- Operators

The basic rules for the syntax of the script are as follows:

- Each line ends with a semicolon (;).
- Variable names start with a dollar sign ($).
- String values are enclosed in single quotation marks (').
- Comments start with a pound sign (#).
- Function calls always specify parameters even if the function uses no parameters.

For example, the following script statement determines today's date and assigns the value to the variable `$`**`<TODAY>`**:

```
$<TODAY> = sysdate();
```

You cannot use variables unless you declare them in the work flow that calls the script.

Related Information

Reference Guide: Data Services Scripting Language [page 1709]

2.9.9.1 To create a script

1. Open the work flow.
2. Click the script icon in the tool palette.
3. Click the location where you want to place the script in the diagram.

   The script icon appears in the diagram.
4. Click the name of the script to open the script editor.
5. Enter the script statements, each followed by a semicolon.

   The following example shows a script that determines the start time from the output of a custom function.

   ```
   AW_StartJob ('NORMAL','DELTA', $G_STIME,$GETIME);
   $GETIME =to_date(
   ```

```
sql('ODS_DS','SELECT to_char(MAX(LAST_UPDATE) ,
\'YYYY-MM-DDD HH24:MI:SS\')
FROM EMPLOYEE'),
'YYYY_MMM_DDD_HH24:MI:SS');
```

Click the function button to include functions in your script.

6. After you complete the script, select *Validation* > *Validate*.

   The software tests your script for syntax errors and displays any errors encountered.
7. Click the *...* button and then *save* to name and save your script.
   The script is saved by default in **<LINK_DIR>**/Data Services/ DataQuality/Samples.

2.9.9.2 Debugging scripts using the print function

The software has a debugging feature that allows you to print:

- The values of variables and parameters during execution
- The execution path followed within a script

You can use the print function to write the values of parameters and variables in a work flow to the trace log. For example, this line in a script:

```
print('The value of parameter $<x>: [$<x>]');
```

produces the following output in the trace log:

```
The following output is being printed via the Print function in <Session <job_name>.
The value of parameter $<x>: <value>
```

Related Information

Reference Guide: Functions and Procedures, print [page 1639]

2.10 Nested Data

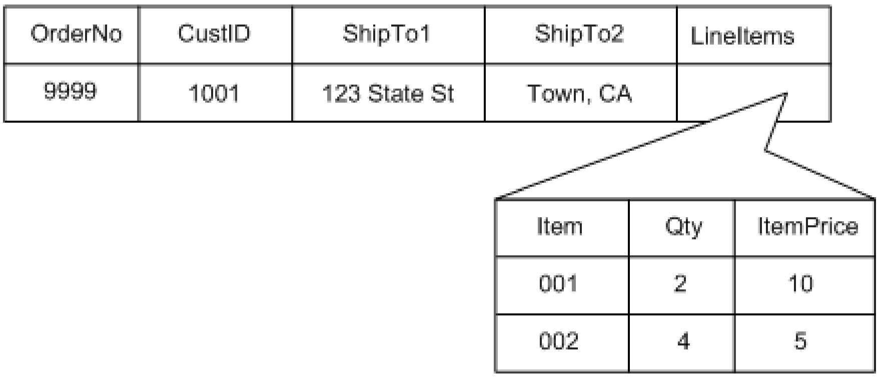
This section discusses nested data and how to use it in the software.

2.10.1 What is nested data?

Real-world data often has hierarchical relationships that are represented in a relational database with master-detail schemas using foreign keys to create the mapping. However, some data sets, such as XML documents and SAP ERP IDocs, handle hierarchical relationships through nested data.

The software maps nested data to a separate schema implicitly related to a single row and column of the parent schema. This mechanism is called Nested Relational Data Modelling (NRDM). NRDM provides a way to view and manipulate hierarchical relationships within data flow sources, targets, and transforms.

Sales orders are often presented using nesting: the line items in a sales order are related to a single header and are represented using a nested schema. Each row of the sales order data set contains a nested line item schema.



OrderNo	CustID	ShipTo1	ShipTo2	LineItems
9999	1001	123 State St	Town, CA	

Item	Qty	ItemPrice
001	2	10
002	4	5

2.10.2 Representing hierarchical data

You can represent the same hierarchical data in several ways. Examples include:

- Multiple rows in a single data set
  Order data set

OrderNo	CustID	ShipTo1	ShipTo2	Item	Qty	ItemPrice
9999	1001	123 State St	Town, CA	001	2	10
9999	1001	123 State St	Town, CA	002	4	5

- Multiple data sets related by a join
  Order header data set

OrderNo	CustID	ShipTo1	ShipTo2
9999	1001	123 State St	Town, CA

Line-item data set

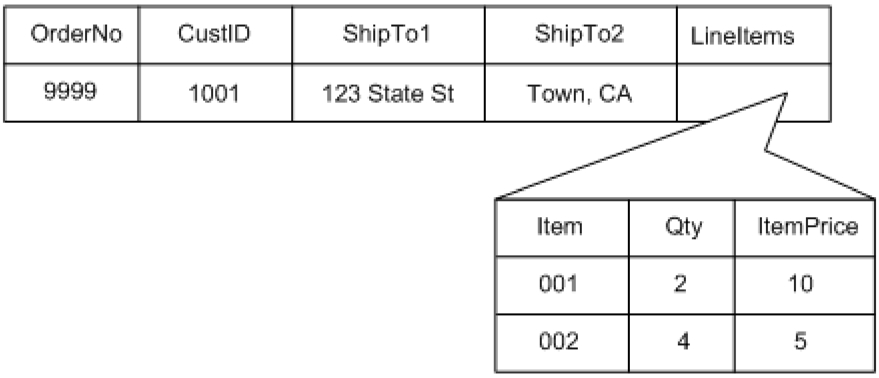
OrderNo	Item	Qty	ItemPrice
9999	001	2	10
9999	002	4	5

WHERE Header.OrderNo=LineItem.OrderNo

- Nested data

Using the nested data method can be more concise (no repeated information), and can scale to present a deeper level of hierarchical complexity. For example, columns inside a nested schema can also contain columns. There is a unique instance of each nested schema for each row at each level of the relationship.

Order data set

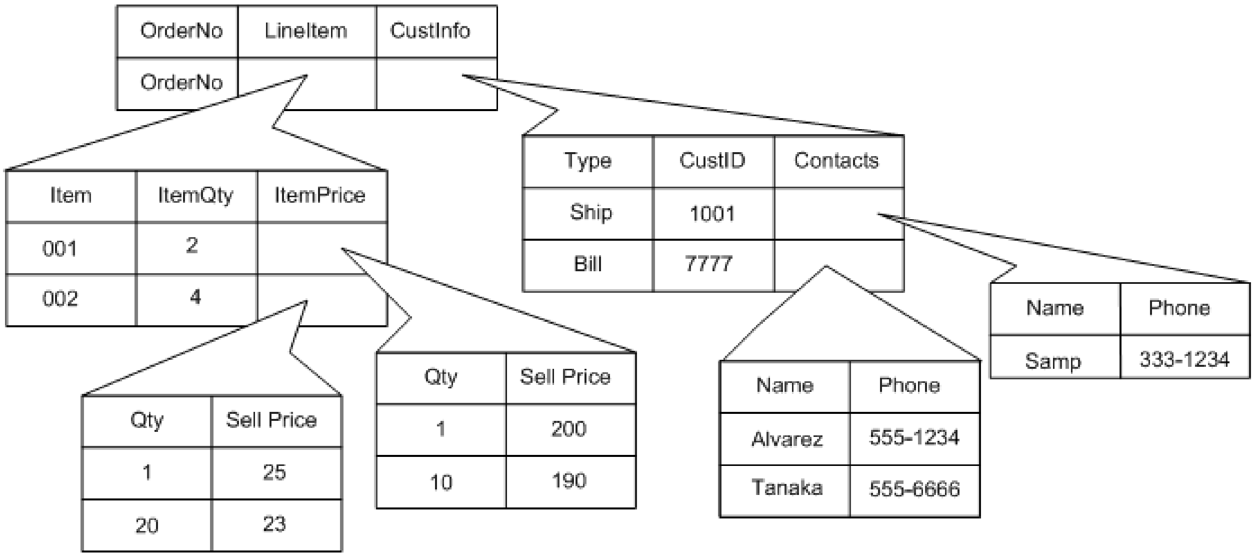


OrderNo	CustID	ShipTo1	ShipTo2	LineItems
9999	1001	123 State St	Town, CA	

Item	Qty	ItemPrice
001	2	10
002	4	5

Generalizing further with nested data, each row at each level can have any number of columns containing nested schemas.

Order data set



OrderNo	LineItem	CustInfo
OrderNo		

Item	ItemQty	ItemPrice
001	2	
002	4	

Qty	Sell Price
1	25
20	23

Qty	Sell Price
1	200
10	190

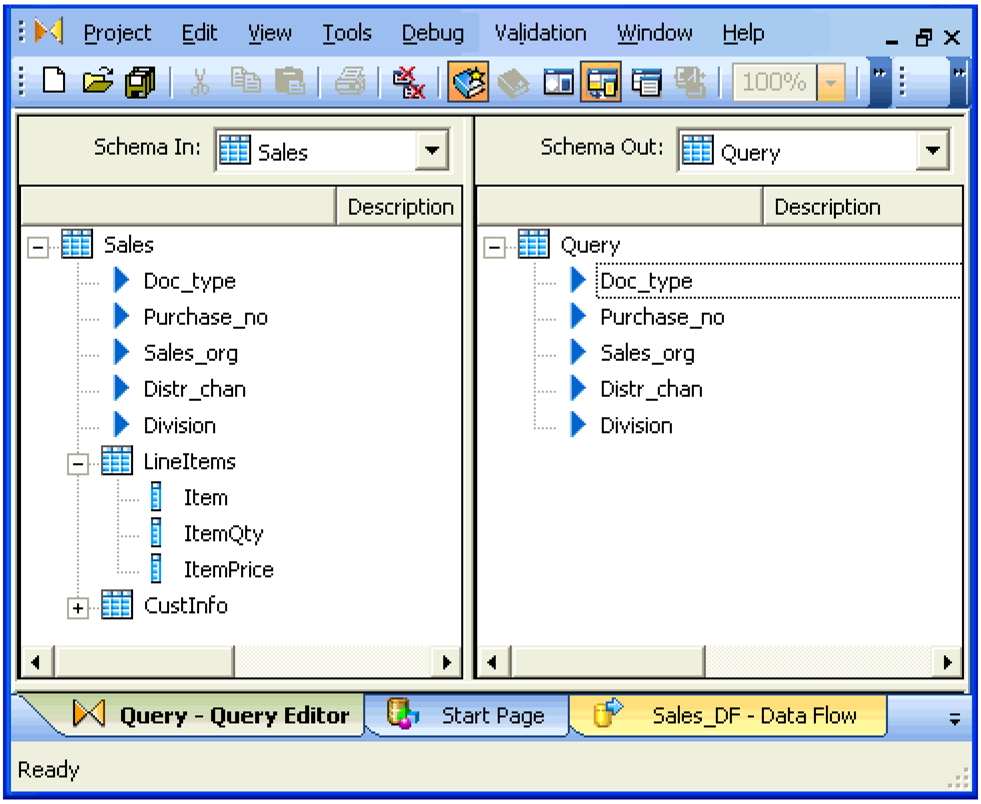
Type	CustID	Contacts
Ship	1001	
Bill	7777	

Name	Phone
Alvarez	555-1234
Tanaka	555-6666

Name	Phone
Samp	333-1234

You can see the structure of nested data in the input and output schemas of sources, targets, and transforms in data flows. Nested schemas appear with a schema icon paired with a plus sign, which indicates that the object contains columns. The structure of the schema shows how the data is ordered.

- Sales is the top-level schema.
- LineItems is a nested schema. The minus sign in front of the schema icon indicates that the column list is open.
- CustInfo is a nested schema with the column list closed.



2.10.3 Formatting XML documents

The software allows you to import and export metadata for XML documents (files or messages), which you can use as sources or targets in jobs. XML documents are hierarchical. Their valid structure is stored in separate format documents.

The format of an XML file or message (.xml) can be specified using either an XML Schema (for example, .xsd) or a document type definition (.dtd).

When you import a format document's metadata, it is structured into the software's internal schema for hierarchical documents which uses the nested relational data model (NRDM).

Related Information

Importing XML Schemas [page 346]
Specifying source options for XML files [page 350]
Mapping optional schemas [page 352]
Using Document Type Definitions (DTDs) [page 353]
Generating DTDs and XML Schemas from an NRDM schema [page 355]

2.10.3.1 Importing XML Schemas

The software supports WC3 XML Schema Specification 1.0.

For an XML document that contains information to place a sales order—order header, customer, and line items—the corresponding XML Schema includes the order structure and the relationship between data.

Message with data

OrderNo	CustID	ShipTo1	ShipTo2	LineItems		
				Item	ItemQty	ItemPrice
9999	1001	123 State St	Town, CA	001	2	10
				002	4	5

Each column in the XML document corresponds to an ELEMENT or attribute definition in the XML schema.

Corresponding XML schema

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Order">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="OrderNo" type="xs:string" />
        <xs:element name="CustID" type="xs:string" />
        <xs:element name="ShipTo1" type="xs:string" />
        <xs:element name="ShipTo2" type="xs:string" />
        <xs:element maxOccurs="unbounded" name="LineItems">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Item" type="xs:string" />
              <xs:element name="ItemQty" type="xs:string" />
              <xs:element name="ItemPrice" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Related Information

Reference Guide: XML schema [page 1008]

2.10.3.1.1 Importing XML schemas

Import the metadata for each XML Schema you use. The object library lists imported XML Schemas in the *Formats* tab.

When importing an XML Schema, the software reads the defined elements and attributes, and then imports the following:

- Document structure
- Namespace
- Table and column names
- Data type of each column
- Content type of each column
- Nested table and column attributes
  While XML Schemas make a distinction between elements and attributes, the software imports and converts them all to nested table and column attributes.

Related Information

Reference Guide: XML schema [page 1008]

2.10.3.1.1.1 To import an XML Schema

1. From the object library, click the *Format* tab.
2. Right-click the *XML Schemas* icon and select *New*.
3. Enter the settings for the XML schemas that you import.

   When importing an XML Schema:

   - Enter the name you want to use for the format in the software.
   - Enter the file name of the XML Schema or its URL address.

     > i Note
     >
     > If your Job Server is on a different computer than the Designer, you cannot use Browse to specify the file path. You must type the path. You can type an absolute path or a relative path, but the Job Server must be able to access it.

   - If the root element name is not unique within the XML Schema, select a name in the *Namespace* drop-down list to identify the imported XML Schema.

     > i Note
     >
     > When you import an XML schema for a real-time web service job, you should use a unique target namespace for the schema. When Data Services generates the WSDL file for a real-time job with a source or target schema that has no target namespace, it adds an automatically generated target namespace to the types section of the XML schema. This can reduce performance because Data

> Services must suppress the namespace information from the web service request during processing, and then reattach the proper namespace information before returning the response to the client.

- In the *Root element name* drop-down list, select the name of the primary node that you want to import. The software only imports elements of the XML Schema that belong to this node or any subnodes.
- If the XML Schema contains recursive elements (element A contains B, element B contains A), specify the number of levels it has by entering a value in the *Circular level* box. This value must match the number of recursive levels in the XML Schema's content. Otherwise, the job that uses this XML Schema will fail.
- You can set the software to import strings as a varchar of any size. Varchar 1024 is the default.

4. Click *OK*.

After you import an XML Schema, you can edit its column properties such as data type using the *General* tab of the Column Properties window. You can also view and edit nested table and column attributes from the Column Properties window.

2.10.3.1.1.2 To view and edit nested table and column attributes for XML Schema

1. From the object library, select the *Formats* tab.
2. Expand the *XML Schema* category.
3. Double-click an XML Schema name.

   The XML Schema Format window appears in the workspace.

   The Type column displays the data types that the software uses when it imports the XML document metadata.
4. Double-click a nested table or column and select *Attributes* to view or edit XML Schema attributes.

Related Information

Reference Guide: XML schema [page 1008]

2.10.3.1.2 Importing abstract types

An XML schema uses abstract types to force substitution for a particular element or type.

- When an element is defined as abstract, a member of the element's substitution group must appear in the instance document.
- When a type is defined as abstract, the instance document must use a type derived from it (identified by the xsi:type attribute).

For example, an abstract element PublicationType can have a substitution group that consists of complex types such as MagazineType, BookType, and NewspaperType.

The default is to select all complex types in the substitution group or all derived types for the abstract type, but you can choose to select a subset.

2.10.3.1.2.1 To limit the number of derived types to import for an abstract type

1. On the Import XML Schema Format window, when you enter the file name or URL address of an XML Schema that contains an abstract type, the *Abstract type* button is enabled.

   For example, the following excerpt from an xsd defines the PublicationType element as abstract with derived types BookType and MagazineType:

   ```
   <xsd:complexType name="PublicationType" abstract="true">
       <xsd:sequence>
           <xsd:element name="Title" type="xsd:string"/>
           <xsd:element name="Author" type="xsd:string" minOccurs="0"
   maxOccurs="unbounded"/>
           <xsd:element name="Date" type="xsd:gYear"/>
       </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="BookType">
       <xsd:complexContent>
           <xsd:extension base="PublicationType">
           <xsd:sequence>
               <xsd:element name="ISBN" type="xsd:string"/>
               <xsd:element name="Publisher" type="xsd:string"/>
           </xsd:sequence>
           </xsd:extension>
       /xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="MagazineType">
           <xsd:complexContent>
               <xsd:restriction base="PublicationType">
                   <xsd:sequence>
                       <xsd:element name="Title" type="xsd:string"/>
                       <xsd:element name="Author" type="xsd:string" minOccurs="0"
   maxOccurs="1"/>
                       <xsd:element name="Date" type="xsd:gYear"/>
                   </xsd:sequence>
               </xsd:restriction>
           </xsd:complexContent>
       </xsd:complexType>
   ```

2. To select a subset of derived types for an abstract type, click the *Abstract type* button and take the following actions:

   a) From the drop-down list on the *Abstract type* box, select the name of the abstract type.

   b) Select the check boxes in front of each derived type name that you want to import.

   c) Click *OK*.

   > i Note
   >
   > When you edit your XML schema format, the software selects all derived types for the abstract type by default. In other words, the subset that you previously selected is not preserved.

2.10.3.1.3 Importing substitution groups

An XML schema uses substitution groups to assign elements to a special group of elements that can be substituted for a particular named element called the head element. The list of substitution groups can have hundreds or even thousands of members, but an application typically only uses a limited number of them. The default is to select all substitution groups, but you can choose to select a subset.

2.10.3.1.3.1 To limit the number of substitution groups to import

1. On the Import XML Schema Format window, when you enter the file name or URL address of an XML Schema that contains substitution groups, the *Substitution Group* button is enabled.

   For example, the following excerpt from an xsd defines the PublicationType element with substitution groups MagazineType, BookType, AdsType, and NewspaperType:

   ```
   <xsd:element name="Publication" type="PublicationType"/>
       <xsd:element name="BookStore">
           <xsd:complexType>
               <xsd:sequence>
                   <xsd:element ref="Publication" maxOccurs="unbounded"/>
               </xsd:sequence>
           </xsd:complexType>
       </xsd:element>
   <xsd:element name="Magazine" type="MagazineType"
   substitutionGroup="Publication"/>
   <xsd:element name="Book" type="BookType" substitutionGroup="Publication"/>
   <xsd:element name="Ads" type="AdsType" substitutionGroup="Publication"/>
   <xsd:element name="Newspaper" type="NewspaperType"
   substitutionGroup="Publication"/>
   ```

2. Click the *Substitution Group* button.
   a) From the drop-down list on the *Substitution group* box, select the name of the substitution group.
   b) Select the check boxes in front of each substitution group name that you want to import.
   c) Click *OK*.

> Note
>
> When you edit your XML schema format, the software selects all elements for the substitution group by default. In other words, the subset that you previously selected is not preserved.

2.10.3.2 Specifying source options for XML files

After you import metadata for XML documents (files or messages), you create a data flow to use the XML documents as sources or targets in jobs.

2.10.3.2.1 Creating a data flow with a source XML file

2.10.3.2.1.1 To create a data flow with a source XML file

1. From the object library, click the *Format* tab.
2. Expand the XML Schema and drag the XML Schema that defines your source XML file into your data flow.
3. Place a query in the data flow and connect the XML source to the input of the query.
4. Double-click the XML source in the work space to open the XML Source File Editor.
5. You must specify the name of the source XML file in the *XML file* text box.

Related Information

Reading multiple XML files at one time [page 351]
Identifying source file names [page 263]
Reference Guide: XML file source [page 954]

2.10.3.2.2 Reading multiple XML files at one time

The software can read multiple files with the same format from a single directory using a single source object.

2.10.3.2.2.1 To read multiple XML files at one time

1. Open the editor for your source XML file.
2. In *XML File* on the Source tab, enter a file name containing a wild card character (* or ?).

   For example:

   `D:\orders\1999????.xml` might read files from the year 1999.

   `D:\orders\*.xml` reads all files with the xml extension from the specified directory.

Related Information

Reference Guide: XML file source [page 954]

2.10.3.2.3 Identifying source file names

You might want to identify the source XML file for each row in your source output in the following situations:

- You specified a wildcard character to read multiple source files at one time.
- You load from a different source file on different days.

2.10.3.2.3.1 To identify the source XML file for each row in the target

1. In the XML Source File Editor, select *Include file name column*, which generates a column DI_FILENAME to contain the name of the source XML file.
2. In the Query editor, map the DI_FILENAME column from Schema In to Schema Out.
3. When you run the job, the target DI_FILENAME column will contain the source XML file name for each row in the target.

2.10.3.3 Mapping optional schemas

You can quickly specify default mapping for optional schemas without having to manually construct an empty nested table for each optional schema in the Query transform. Also, when you import XML schemas (either through DTDs or XSD files), the software automatically marks nested tables as optional if the corresponding option was set in the DTD or XSD file. The software retains this option when you copy and paste schemas into your Query transforms.

This feature is especially helpful when you have very large XML schemas with many nested levels in your jobs. When you make a schema column optional and do not provide mapping for it, the software instantiates the empty nested table when you run the job.

While a schema element is marked as optional, you can still provide a mapping for the schema by appropriately programming the corresponding sub-query block with application logic that specifies how the software should produce the output. However, if you modify any part of the sub-query block, the resulting query block must be complete and conform to normal validation rules required for a nested query block. You must map any output schema not marked as optional to a valid nested query block. The software generates a NULL in the corresponding PROJECT list slot of the ATL for any optional schema without an associated, defined sub-query block.

2.10.3.3.1 To make a nested table "optional"

1. Right-click a nested table and select *Optional* to toggle it on. To toggle it off, right-click the nested table again and select *Optional* again.
2. You can also right-click a nested table and select *Properties*, and then open the Attributes tab and set the *Optional Table* attribute value to *yes* or *no*. Click *Apply* and *OK* to set.

Note

If the Optional Table value is something other than yes or no, the nested table cannot be marked as optional.

When you run a job with a nested table set to optional and you have nothing defined for any columns and nested tables beneath that table, the software generates special ATL and does not perform user interface validation for this nested table.

Example:

```
CREATE NEW Query ( EMPNO int KEY ,

ENAME varchar(10),
JOB varchar (9)
NT1 al_nested_table ( DEPTNO int KEY ,
DNAME varchar (14),
NT2 al_nested_table (C1 int) ) SET("Optional
Table" = 'yes') )
AS SELECT EMP.EMPNO, EMP.ENAME, EMP.JOB,

NULL FROM EMP, DEPT;
```

Note

You cannot mark top-level schemas, unnested tables, or nested tables containing function calls optional.

2.10.3.4 Using Document Type Definitions (DTDs)

The format of an XML document (file or message) can be specified by a document type definition (DTD). The DTD describes the data contained in the XML document and the relationships among the elements in the data.

For an XML document that contains information to place a sales order—order header, customer, and line items—the corresponding DTD includes the order structure and the relationship between data.

Message with data

OrderNo	CustID	ShipTo1	ShipTo2	LineItems		
9999	1001	123 State St	Town, CA	Item	ItemQty	ItemPrice
				001	2	10
				002	4	5

Each column in the XML document corresponds to an ELEMENT definition.

Corresponding DTD Definition

```
<?xml encoding="UTF-8"?>
<!ELEMENT Order (OrderNo, CustID, ShipTo1, ShipTo2, LineItems+)>
<!ELEMENT OrderNo (#PCDATA)>
<!ELEMENT CustID (#PCDATA)>
<!ELEMENT ShipTo1 (#PCDATA)>
<!ELEMENT ShipTo2 (#PCDATA)>
<!ELEMENT LineItems (Item, ItemQty, ItemPrice)>
<!ELEMENT Item (#PCDATA)>
<!ELEMENT ItemQty (#PCDATA)>
<!ELEMENT ItemPrice (#PCDATA)>
```

Import the metadata for each DTD you use. The object library lists imported DTDs in the *Formats* tab.

You can import metadata from either an existing XML file (with a reference to a DTD) or DTD file. If you import the metadata from an XML file, the software automatically retrieves the DTD for that XML file.

When importing a DTD, the software reads the defined elements and attributes. The software ignores other parts of the definition, such as text and comments. This allows you to modify imported XML data and edit the data type as needed.

Related Information

Reference Guide: DTD [page 906]

2.10.3.4.1 To import a DTD or XML Schema format

1. From the object library, click the *Format* tab.
2. Right-click the *DTDs* icon and select *New*.
3. Enter settings into the Import DTD Format window:
    - In the *DTD definition name* box, enter the name that you want to give the imported DTD format in the software.
    - Enter the file that specifies the DTD you want to import.

      Note

      If your Job Server is on a different computer than the Designer, you cannot use Browse to specify the file path. You must type the path. You can type an absolute path or a relative path, but the Job Server must be able to access it.

    - If importing an XML file, select *XML* for the *File type* option. If importing a DTD file, select the *DTD* option.
    - In the *Root element name* box, select the name of the primary node that you want to import. The software only imports elements of the DTD that belong to this node or any subnodes.
    - If the DTD contains recursive elements (element A contains B, element B contains A), specify the number of levels it has by entering a value in the *Circular level* box. This value must match the number of recursive levels in the DTD's content. Otherwise, the job that uses this DTD will fail.

- ○ You can set the software to import strings as a varchar of any size. Varchar 1024 is the default.

4. Click *OK*.

After you import a DTD, you can edit its column properties, such as data type, using the *General* tab of the Column Properties window. You can also view and edit DTD nested table and column attributes from the Column Properties window.

2.10.3.4.2 To view and edit nested table and column attributes for DTDs

1. From the object library, select the *Formats* tab.
2. Expand the *DTDs* category.
3. Double-click a DTD name.

   The DTD Format window appears in the workspace.
4. Double-click a nested table or column.

   The Column Properties window opens.
5. Select the *Attributes* tab to view or edit DTD attributes.

2.10.3.5 Generating DTDs and XML Schemas from an NRDM schema

You can right-click any schema from within a query editor in the Designer and generate a DTD or an XML Schema that corresponds to the structure of the selected schema (either NRDM or relational).

This feature is useful if you want to stage data to an XML file and subsequently read it into another data flow.

1. Generate a DTD/XML Schema.
2. Use the DTD/XML Schema to setup an XML format.
3. Use the XML format to set up an XML source for the staged file.

The DTD/XML Schema generated will be based on the following information:

- Columns become either elements or attributes based on whether the XML Type attribute is set to ATTRIBUTE or ELEMENT.
- If the Required attribute is set to NO, the corresponding element or attribute is marked optional.
- Nested tables become intermediate elements.
- The Native Type attribute is used to set the type of the element or attribute.
- While generating XML Schemas, the MinOccurs and MaxOccurs values are set based on the Minimum Occurrence and Maximum Occurrence attributes of the corresponding nested table.

No other information is considered while generating the DTD or XML Schema.

Related Information

Reference Guide: DTD [page 906]
Reference Guide: XML schema [page 1008]

2.10.4 Operations on nested data

This section discusses the operations that you can perform on nested data.

2.10.4.1 Overview of nested data and the Query transform

With relational data, a Query transform allows you to execute a SELECT statement. The mapping between input and output schemas defines the project list for the statement. When working with nested data, the Query transform provides an interface to perform SELECT statements at each level of the relationship that you define in the output schema.

You use the Query transform to manipulate nested data. If you want to extract only part of the nested data, you can use the XML_Pipeline transform.

Without nested schemas, the Query transform assumes that the FROM clause in the SELECT statement contains the data sets that are connected as inputs to the query object. When working with nested data, you must explicitly define the FROM clause in a query. The software assists by setting the top-level inputs as the default FROM clause values for the top-level output schema.

The other SELECT statement elements defined by the query work the same with nested data as they do with flat data. However, because a SELECT statement can only include references to relational data sets, a query that includes nested data includes a SELECT statement to define operations for each parent and child schema in the output.

The Query Editor contains a tab for each clause of the query:

- SELECT provides an option to specify distinct rows to output (discarding any identical duplicate rows).
- FROM lists all input schemas and allows you to specify join pairs and conditions.

The parameters you enter for the following tabs apply only to the current schema (displayed in the Schema Out text box at the top right of the Query Editor):

- WHERE
- GROUP BY
- ORDER BY

Related Information

Query Editor [page 311]
Reference Guide: XML_Pipeline [page 1119]

2.10.4.2 FROM clause construction

The FROM clause is located at the bottom of the FROM tab. It automatically populates with the information included in the Input Schema(s) section at the top, and the Join Pairs section in the middle of the tab. You can change the FROM clause by changing the selected schema in the Input Schema(s) area and the Join Pairs section.

Schemas selected in the Input Schema(s) section (and reflected in the FROM clause), including columns containing nested schemas, are available to be included in the output.

When you include more than one schema in the Input Schema(s) section (by selecting the *From* check box), you can specify join pairs and join conditions as well as enter join rank and cache for each input schema.

FROM clause descriptions and the behavior of the query are exactly the same with nested data as with relational data. The current schema allows you to distinguish multiple SELECT statements from each other within a single query. However, because the SELECT statements are dependent upon each other, and because the user interface makes it easy to construct arbitrary data sets, determining the appropriate FROM clauses for multiple levels of nesting can be complex.

A FROM clause can contain:

- Any top-level schema from the input
- Any schema that is a column of a schema in the FROM clause of the parent schema
- Any join conditions from the join pairs

The FROM clause forms a path that can start at any level of the output. The first schema in the path must always be a top-level schema from the input.

The data that a SELECT statement from a lower schema produces differs depending on whether or not a schema is included in the FROM clause at the top-level.

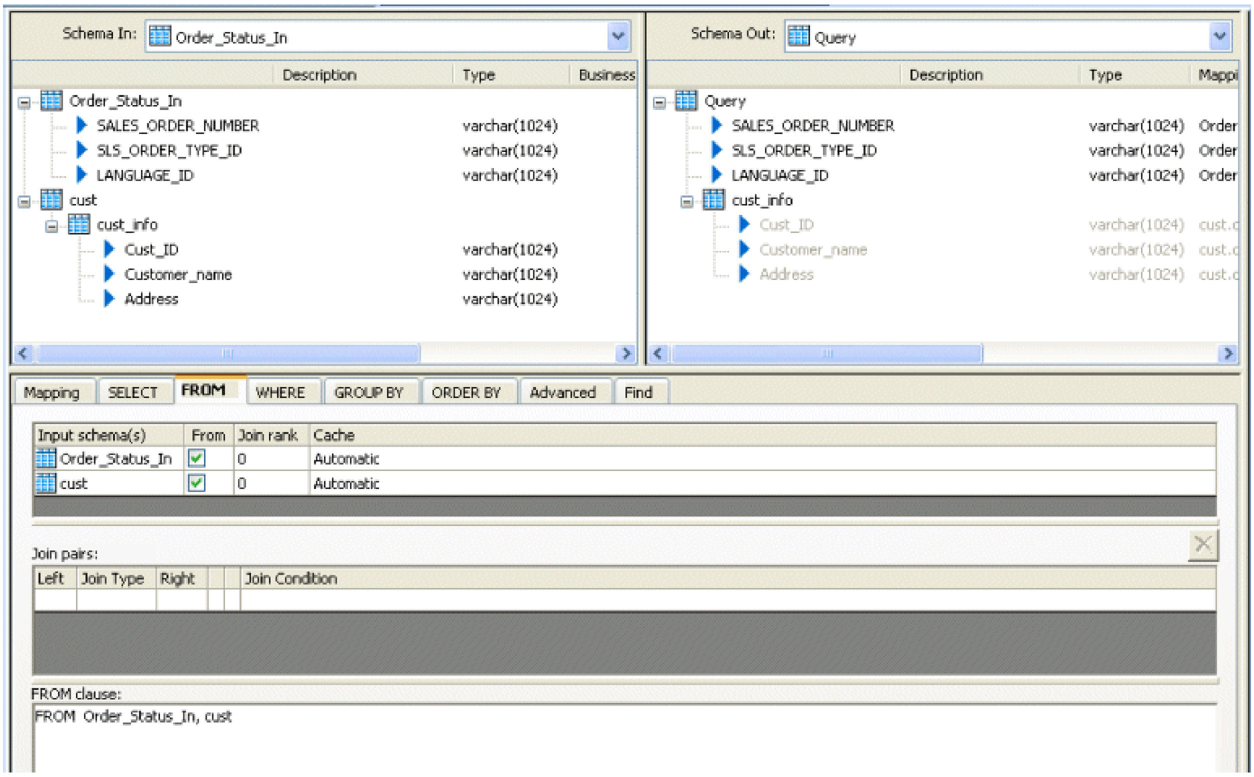
The next two examples use the sales order data set to illustrate scenarios where FROM clause values change the data resulting from the query.

Related Information

To modify the output schema contents [page 311]

2.10.4.2.1 Example: FROM clause includes all top-level inputs

To include detailed customer information for all of the orders in the output, join the Order_Status_In schema at the top level with the Cust schema. Include both input schemas at the top level in the FROM clause to produce the appropriate data. When you select both input schemas in the Input schema(s) area of the FROM tab, they automatically appear in the FROM clause.



Schema In: Order_Status_In

	Description	Type	Business
Order_Status_In			
SALES_ORDER_NUMBER		varchar(1024)	
SLS_ORDER_TYPE_ID		varchar(1024)	
LANGUAGE_ID		varchar(1024)	
cust			
cust_info			
Cust_ID		varchar(1024)	
Customer_name		varchar(1024)	
Address		varchar(1024)	

Schema Out: Query

	Description	Type	Mappi
Query			
SALES_ORDER_NUMBER		varchar(1024)	Order
SLS_ORDER_TYPE_ID		varchar(1024)	Order
LANGUAGE_ID		varchar(1024)	Order
cust_info			
Cust_ID		varchar(1024)	cust.c
Customer_name		varchar(1024)	cust.c
Address		varchar(1024)	cust.c

Input schema(s)	From	Join rank	Cache
Order_Status_In	☑	0	Automatic
cust	☑	0	Automatic

Join pairs:

Left	Join Type	Right			Join Condition

FROM clause:

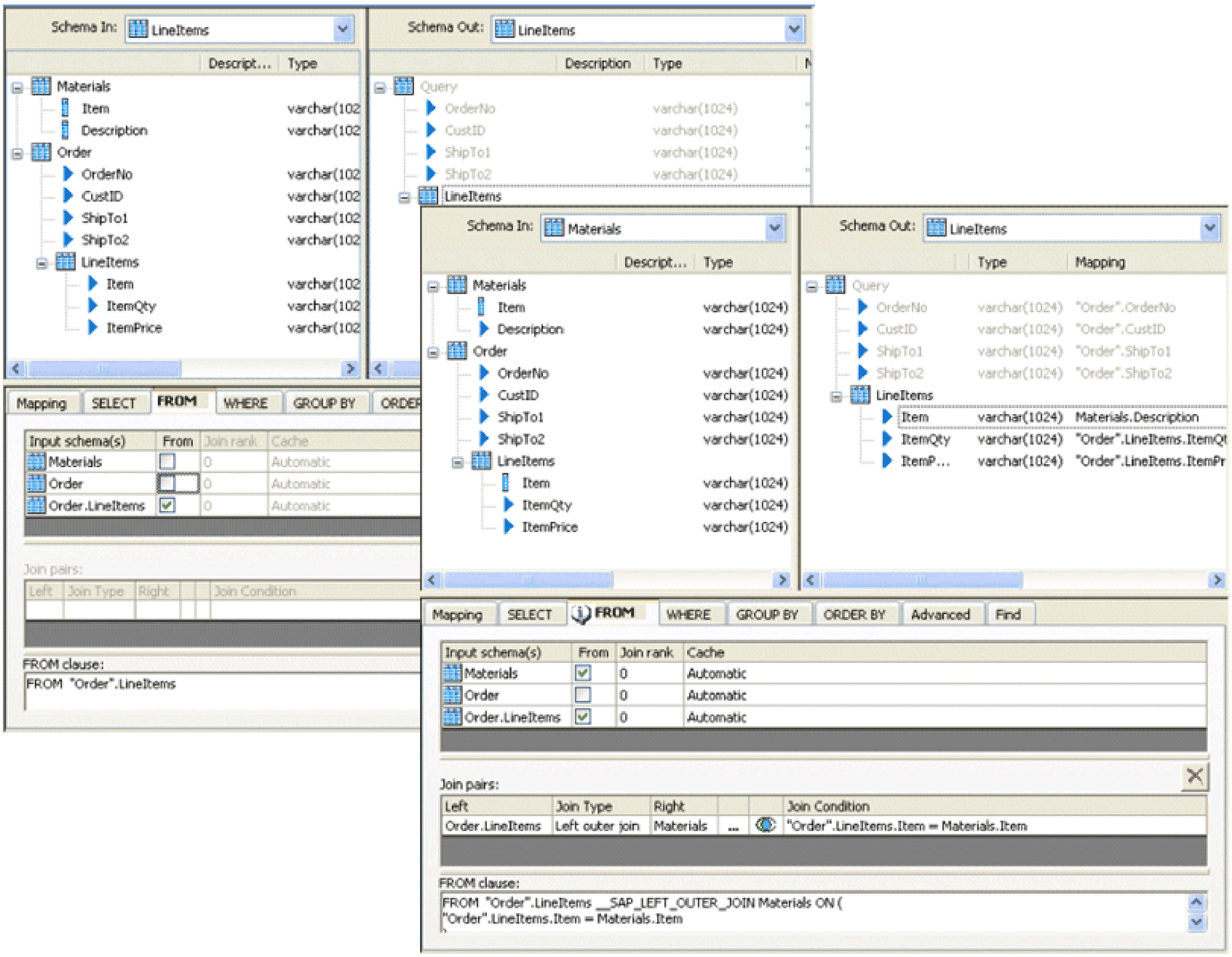
```
FROM Order_Status_In, cust
```

Observe the following points:

- The Input schema(s) table in the FROM tab includes the two top-level schemas Order_Status_In and Cust (this is also reflected in the FROM clause).
- The Schema Out pane shows the nested schema, cust_info, and the columns Cust_ID, Customer_name, and Address.

2.10.4.2.2 Example: Lower level FROM clause contains top-level input

Suppose you want the detailed information from one schema to appear for each row in a lower level of another schema. For example, the input includes a top-level Materials schema and a nested LineItems schema, and you want the output to include detailed material information for each line item. The graphic below illustrates how to set this up in the Designer.



Schema In: LineItems

	Descript...	Type
Materials		
Item		varchar(102
Description		varchar(102
Order		
OrderNo		varchar(102
CustID		varchar(102
ShipTo1		varchar(102
ShipTo2		varchar(102
LineItems		
Item		varchar(102
ItemQty		varchar(102
ItemPrice		varchar(102

Schema Out: LineItems

	Description	Type
Query		
OrderNo		varchar(1024)
CustID		varchar(1024)
ShipTo1		varchar(1024)
ShipTo2		varchar(1024)
LineItems		

Input schema(s)	From	Join rank	Cache
Materials	☐	0	Automatic
Order	☐	0	Automatic
Order.LineItems	☑	0	Automatic

Join pairs:

Left	Join Type	Right		Join Condition

FROM clause:
FROM "Order".LineItems

Schema In: Materials

	Descript...	Type
Materials		
Item		varchar(1024)
Description		varchar(1024)
Order		
OrderNo		varchar(1024)
CustID		varchar(1024)
ShipTo1		varchar(1024)
ShipTo2		varchar(1024)
LineItems		
Item		varchar(1024)
ItemQty		varchar(1024)
ItemPrice		varchar(1024)

Schema Out: LineItems

	Type	Mapping
Query		
OrderNo	varchar(1024)	"Order".OrderNo
CustID	varchar(1024)	"Order".CustID
ShipTo1	varchar(1024)	"Order".ShipTo1
ShipTo2	varchar(1024)	"Order".ShipTo2
LineItems		
Item	varchar(1024)	Materials.Description
ItemQty	varchar(1024)	"Order".LineItems.ItemQt
ItemP...	varchar(1024)	"Order".LineItems.ItemPr

Input schema(s)	From	Join rank	Cache
Materials	☑	0	Automatic
Order	☐	0	Automatic
Order.LineItems	☑	0	Automatic

Join pairs:

Left	Join Type	Right		Join Condition
Order.LineItems	Left outer join	Materials	...	"Order".LineItems.Item = Materials.Item

FROM clause:
FROM "Order".LineItems __SAP_LEFT_OUTER_JOIN Materials ON (
"Order".LineItems.Item = Materials.Item

The example on the left shows the following setup:

- The Input Schema area in the FROM tab shows the nested schema LineItems selected.
- The FROM tab shows the FROM clause "FROM "Order".LineItems".

The example on the right shows the following setup:

- The Materials.Description schema is mapped to LineItems.Item output schema.
- The Input schema(s) Materials and Order.LineItems are selected in the Input Schema area in the FROM tab (the From column has a check mark).
- A Join Pair is created joining the nested Order.LineItems schema with the top-level Materials schema using a left outer join type.
- A Join Condition is added where the Item field under the nested schema LineItems is equal to the Item field in the top-level Materials schema.

The resulting FROM clause:

```
"Order".LineItems.Item = Materials.Item
```

2.10.4.3 Nesting columns

When you nest rows of one schema inside another, the data set produced in the nested schema is the result of a query against the first one using the related values from the second one.

For example, if you have sales-order information in a header schema and a line-item schema, you can nest the line items under the header schema. The line items for a single row of the header schema are equal to the results of a query including the order number:

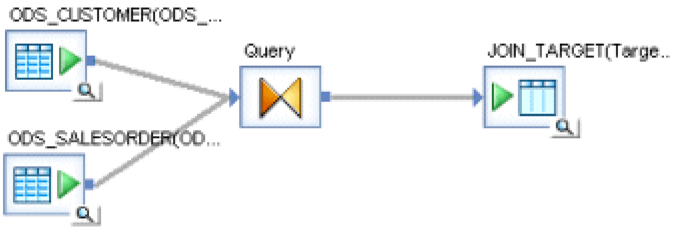
```
SELECT * FROM LineItems

WHERE Header.OrderNo = LineItems.OrderNo
```

You can use a query transform to construct a nested data set from relational data. When you indicate the columns included in the nested schema, specify the query used to define the nested data set for each row of the parent schema.

2.10.4.3.1 To construct a nested data set

Follow the steps below to set up a nested data set.

1. Create a data flow with the input sources that you want to include in the nested data set.
2. Place a Query transform and a target table in the data flow. Connect the sources to the input of the query.



3. Open the Query transform and set up the select list, FROM clause, and WHERE clause to describe the SELECT statement that the query executes to determine the top-level data set.
   - *Select list*: Map the input schema items to the output schema by dragging the columns from the input schema to the output schema. You can also include new columns or include mapping expressions for the columns.
   - *FROM clause*: Include the input sources in the list on the *FROM* tab, and include any joins and join conditions required to define the data.
   - *WHERE clause*: Include any filtering required to define the data set for the top-level output.
4. Create a new schema in the output.

   Right-click in the Schema Out area of the Query Editor, choose *New Output Schema*, and name the new schema. A new schema icon appears in the output, nested under the top-level schema.

   You can also drag an entire schema from the input to the output.
5. Change the current output schema to the nested schema by right-clicking the nested schema and selecting *Make Current*.

The Query Editor changes to display the new current schema.

6. Indicate the FROM clause, select list, and WHERE clause to describe the SELECT statement that the query executes to determine the top-level data set.
   - *FROM clause*: If you created a new output schema, you need to drag schemas from the input to populate the FROM clause. If you dragged an existing schema from the input to the top-level output, that schema is automatically mapped and listed in the From tab.
   - *Select list*: Only columns are available that meet the requirements for the FROM clause.
   - *WHERE clause*: Only columns are available that meet the requirements for the FROM clause.
7. If the output requires it, nest another schema at this level.

   Repeat steps 4 through 6 in this current schema for as many nested schemas that you want to set up.
8. If the output requires it, nest another schema under the top level.

   Make the top-level schema the current schema.

Related Information

Query Editor [page 311]
FROM clause construction [page 357]
To modify the output schema contents [page 311]

2.10.4.4 Using correlated columns in nested data

Correlation allows you to use columns from a higher-level schema to construct a nested schema. In a nested-relational model, the columns in a nested schema are implicitly related to the columns in the parent row. To take advantage of this relationship, you can use columns from the parent schema in the construction of the nested schema. The higher-level column is a correlated column.

Including a correlated column in a nested schema can serve two purposes:

- The correlated column is a key in the parent schema. Including the key in the nested schema allows you to maintain a relationship between the two schemas after converting them from the nested data model to a relational model.
- The correlated column is an attribute in the parent schema. Including the attribute in the nested schema allows you to use the attribute to simplify correlated queries against the nested data.

To include a correlated column in a nested schema, you do not need to include the schema that includes the column in the FROM clause of the nested schema.

2.10.4.4.1 To used a correlated column in a nested schema

1. Create a data flow with a source that includes a parent schema with a nested schema.

For example, the source could be an order header schema that has a LineItems column that contains a nested schema.

2. Connect a query to the output of the source.
3. In the query editor, copy all columns of the parent schema to the output.

   In addition to the top-level columns, the software creates a column called LineItems that contains a nested schema that corresponds to the LineItems nested schema in the input.
4. Change the current schema to the LineItems schema.
5. Include a correlated column in the nested schema.

   Correlated columns can include columns from the parent schema and any other schemas in the FROM clause of the parent schema.

   For example, drag the OrderNo column from the Header schema into the LineItems schema. Including the correlated column creates a new output column in the LineItems schema called OrderNo and maps it to the Order.OrderNo column. The data set created for LineItems includes all of the LineItems columns and the OrderNo.

   If the correlated column comes from a schema other than the immediate parent, the data in the nested schema includes only the rows that match both the related values in the current row of the parent schema and the value of the correlated column.

You can always remove the correlated column from the lower-level schema in a subsequent query transform.

Related Information

Query Editor [page 311]

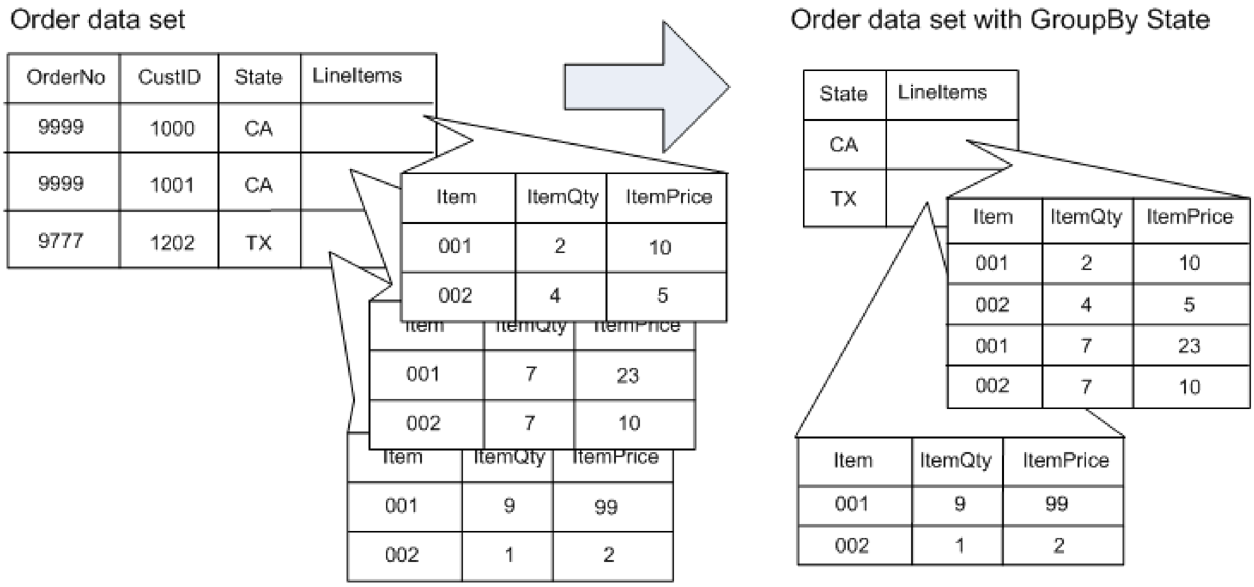
2.10.4.5 Distinct rows and nested data

The *Distinct rows* option in Query transforms removes any duplicate rows at the top level of a join. This is particularly useful to avoid cross products in joins that produce nested output.

2.10.4.6 Grouping values across nested schemas

When you specify a Group By clause for a schema with a nested schema, the grouping operation combines the nested schemas for each group.

For example, to assemble all the line items included in all the orders for each state from a set of orders, you can set the Group By clause in the top level of the data set to the state column (Order.State) and create an output schema that includes State column (set to Order.State) and LineItems nested schema.



Order data set

OrderNo	CustID	State	LineItems
9999	1000	CA	
9999	1001	CA	
9777	1202	TX	

Item	ItemQty	ItemPrice
001	2	10
002	4	5

Item	ItemQty	ItemPrice
001	7	23
002	7	10

Item	ItemQty	ItemPrice
001	9	99
002	1	2

Order data set with GroupBy State

State	LineItems
CA	
TX	

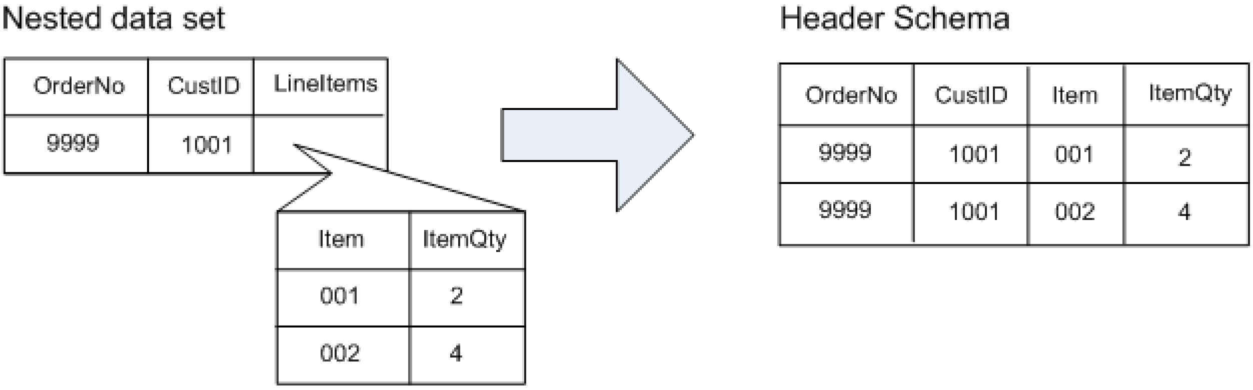
Item	ItemQty	ItemPrice
001	2	10
002	4	5
001	7	23
002	7	10

Item	ItemQty	ItemPrice
001	9	99
002	1	2

The result is a set of rows (one for each state) that has the State column and the LineItems nested schema that contains all the LineItems for all the orders for that state.

2.10.4.7 Unnesting nested data

Loading a data set that contains nested schemas into a relational (non-nested) target requires that the nested rows be unnested. For example, a sales order may use a nested schema to define the relationship between the order header and the order line items. To load the data into relational schemas, the multi-level must be unnested. Unnesting a schema produces a cross-product of the top-level schema (parent) and the nested schema (child).



Nested data set

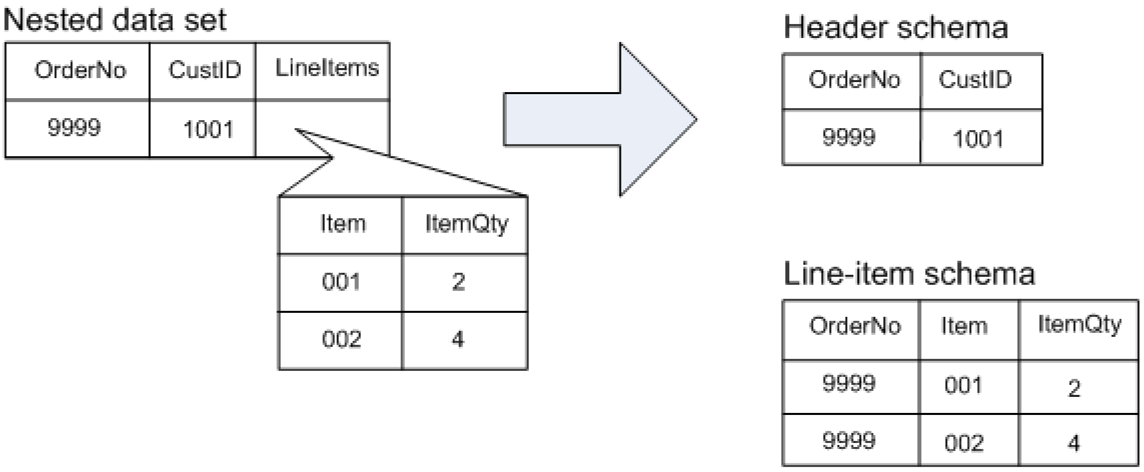
OrderNo	CustID	LineItems
9999	1001	

Item	ItemQty
001	2
002	4

Header Schema

OrderNo	CustID	Item	ItemQty
9999	1001	001	2
9999	1001	002	4

It is also possible that you would load different columns from different nesting levels into different schemas. A sales order, for example, may be flattened so that the order number is maintained separately with each line item and the header and line item information loaded into separate schemas.



Nested data set

OrderNo	CustID	LineItems
9999	1001	

Item	ItemQty
001	2
002	4

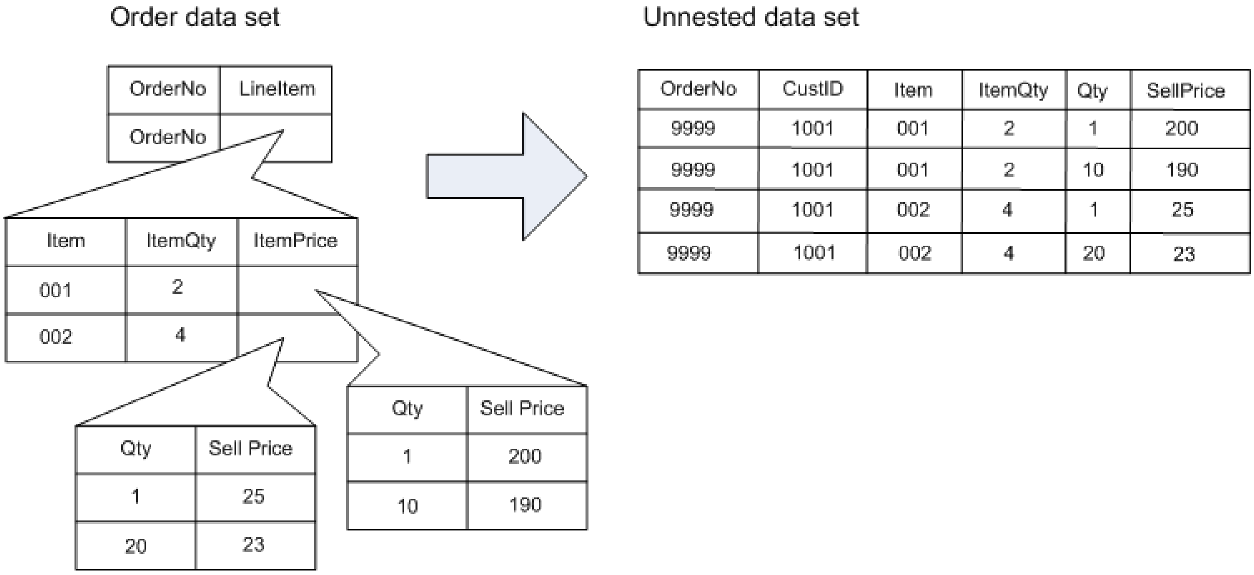
Header schema

OrderNo	CustID
9999	1001

Line-item schema

OrderNo	Item	ItemQty
9999	001	2
9999	002	4

The software allows you to unnest any number of nested schemas at any depth. No matter how many levels are involved, the result of unnesting schemas is a cross product of the parent and child schemas. When more than one level of unnesting occurs, the inner-most child is unnested first, then the result—the cross product of the parent and the inner-most child—is then unnested from its parent, and so on to the top-level schema.



Order data set

OrderNo	LineItem
OrderNo	

Item	ItemQty	ItemPrice
001	2	
002	4	

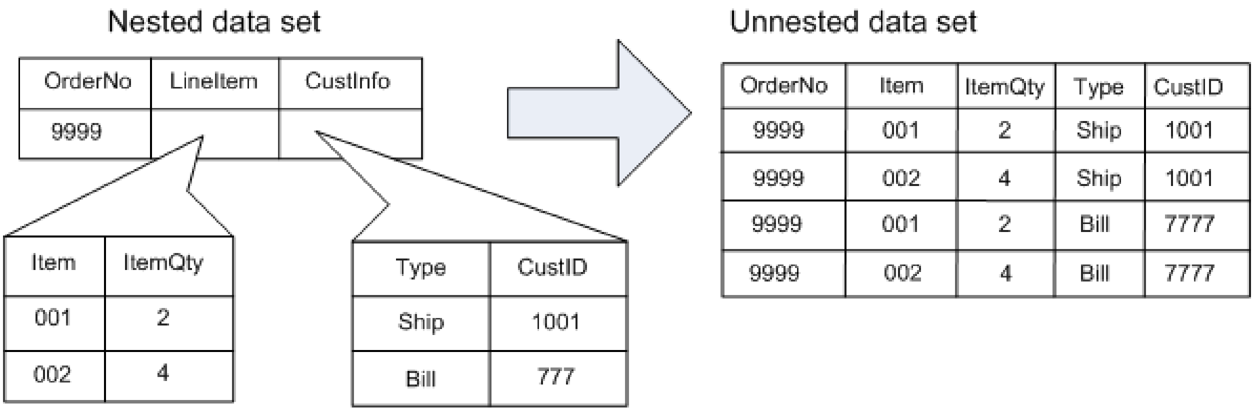
Qty	Sell Price
1	25
20	23

Qty	Sell Price
1	200
10	190

Unnested data set

OrderNo	CustID	Item	ItemQty	Qty	SellPrice
9999	1001	001	2	1	200
9999	1001	001	2	10	190
9999	1001	002	4	1	25
9999	1001	002	4	20	23

Unnesting all schemas (cross product of all data) might not produce the results that you intend. For example, if an order includes multiple customer values such as ship-to and bill-to addresses, flattening a sales order by unnesting customer and line-item schemas produces rows of data that might not be useful for processing the order.



Nested data set

OrderNo	LineItem	CustInfo
9999		

Item	ItemQty
001	2
002	4

Type	CustID
Ship	1001
Bill	777

Unnested data set

OrderNo	Item	ItemQty	Type	CustID
9999	001	2	Ship	1001
9999	002	4	Ship	1001
9999	001	2	Bill	7777
9999	002	4	Bill	7777

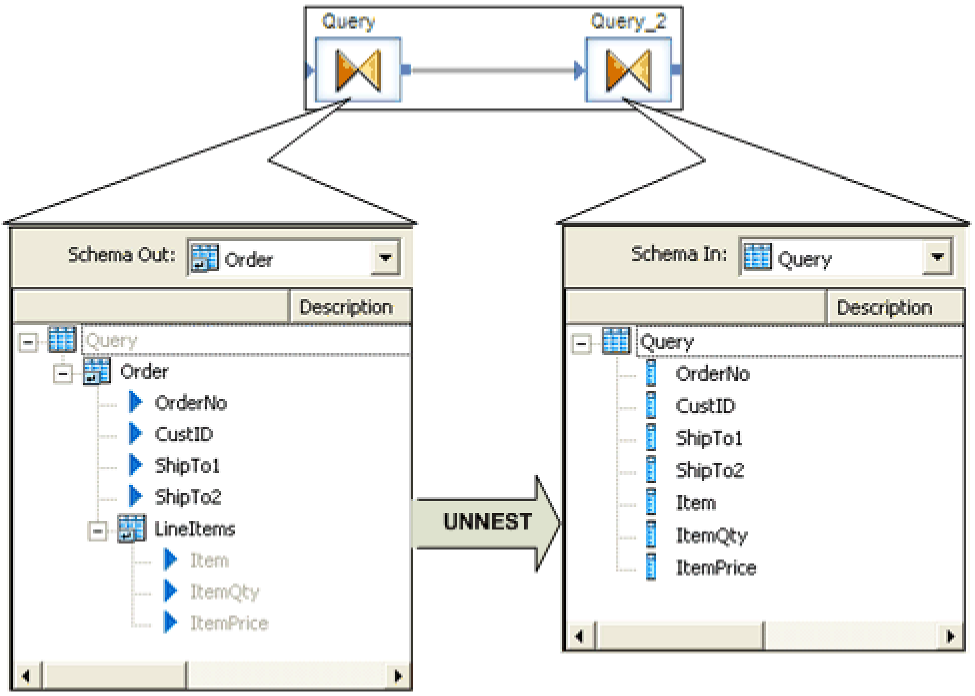
2.10.4.7.1 To unnest nested data

1. Create the output that you want to unnest in the output schema of a query.

   Data for unneeded columns or schemas might be more difficult to filter out after the unnesting operation. You can use the Cut command to remove columns or schemas from the top level; to remove nested schemas or columns inside nested schemas, make the nested schema the current schema, and then cut the unneeded columns or nested columns.

2. For each of the nested schemas that you want to unnest, right-click the schema name and choose *Unnest*.

   The output of the query (the input to the next step in the data flow) includes the data in the new relationship, as the following diagram shows.



2.10.4.8 Transforming lower levels of nested data

Nested data included in the input to transforms (with the exception of a Query or XML_Pipeline transform) passes through the transform without being included in the transform's operation. Only the columns at the first level of the input data set are available for subsequent transforms.

2.10.4.8.1 To transform values in lower levels of nested schemas

1. Take one of the following actions to obtain the nested data:
   - Use a Query transform to unnest the data.
   - Use an XML_Pipeline transform to select portions of the nested data.
   - Perform the transformation.
2. Nest the data again to reconstruct the nested relationships.

Related Information

Unnesting nested data [page 363]

Reference Guide: XML_Pipeline [page 1119]

2.10.5 XML extraction and parsing for columns

In addition to extracting XML message and file data, representing it as NRDM data during transformation, and then loading it to an XML message or file, you can also use the software to extract XML data stored in a source table or flat file column, transform it as NRDM data, and then load it to a target or flat file column.

More and more database vendors allow you to store XML in one column. The field is usually a varchar, long, or clob. The software's XML handling capability also supports reading from and writing to such fields. The software provides four functions to support extracting from and loading to columns:

- extract_from_xml
- load_to_xml
- long_to_varchar
- varchar_to_long

The extract_from_xml function gets the XML content stored in a single column and builds the corresponding NRDM structure so that the software can transform it. This function takes varchar data only.

To enable extracting and parsing for columns, data from long and clob columns must be converted to varchar before it can be transformed by the software.

- The software converts a clob data type input to varchar if you select the *Import unsupported data types as VARCHAR of size* option when you create a database datastore connection in the Datastore Editor.

- If your source uses a long data type, use the long_to_varchar function to convert data to varchar.

> **Note**
>
> The software limits the size of the XML supported with these methods to 100K due to the current limitation of its varchar data type. There are plans to lift this restriction in the future.

The function load_to_xml generates XML from a given NRDM structure in the software, then loads the generated XML to a varchar column. If you want a job to convert the output to a long column, use the varchar_to_long function, which takes the output of the load_to_xml function as input.

2.10.5.1 Sample scenarios

The following scenarios describe how to use functions to extract XML data from a source column and load it into a target column.

Related Information

Extracting XML data from a column into the software [page 367]
Loading XML data into a column of the data type long [page 369]
Extracting data quality XML strings using extract_from_xml function [page 370]

2.10.5.1.1 Extracting XML data from a column into the software

This scenario uses long_to_varchar and extract_from_xml functions to extract XML data from a column with data of the type long.

1. First, assume you have previously performed the following steps:
   a) Imported an Oracle table that contains a column named *Content* with the data type long, which contains XML data for a purchase order.
   b) Imported the XML Schema PO.xsd, which provides the format for the XML data, into the repository.
   c) Created a Project, a job, and a data flow for your design.
   d) Opened the data flow and dropped the source table with the column named *content* in the data flow.
2. From this point:
   a) Create a query with an output column of data type varchar, and make sure that its size is big enough to hold the XML data.
   b) Name this output column *content*.
   c) In the Map section of the query editor, open the Function Wizard, select the *Conversion* function type, then select the *long_to_varchar* function and configure it by entering its parameters.

```
long_to_varchar(content, 4000)
```

The second parameter in this function (4000 in this case) is the maximum size of the XML data stored in the table column. Use this parameter with caution. If the size is not big enough to hold the maximum XML data for the column, the software will truncate the data and cause a runtime error. Conversely, do not enter a number that is too big, which would waste computer memory at runtime.

d) In the query editor, map the source table column to a new output column.

e) Create a second query that uses the function extract_from_xml to extract the XML data.

To invoke the function extract_from_xml, right-click the current context in the query, choose *New Function Call*.

When the Function Wizard opens, select Conversion and extract_from_xml.

> ℹ Note
>
> You can only use the extract_from_xml function in a new function call. Otherwise, this function is not displayed in the function wizard.

f) Enter values for the input parameters.

- The first is the XML column name. Enter *content*, which is the output column in the previous query that holds the XML data
- The second parameter is the DTD or XML Schema name. Enter the name of the purchase order schema (in this case *PO*)
- The third parameter is Enable validation. Enter `1` if you want the software to validate the XML with the specified Schema. Enter `0` if you do not.

g) Click *Next*.

h) For the function, select a column or columns that you want to use on output.

Imagine that this purchase order schema has five top-level elements: orderDate, shipTo, billTo, comment, and items. You can select any number of the top-level columns from an XML schema, which include either scalar or NRDM column data. The return type of the column is defined in the schema. If the function fails due to an error when trying to produce the XML output, the software returns NULL for scalar columns and empty nested tables for NRDM columns.

The extract_from_xml function also adds two columns:

- AL_ERROR_NUM — returns error codes: `0` for success and a non-zero integer for failures
- AL_ERROR_MSG — returns an error message if AL_ERROR_NUM is not `0`. Returns NULL if AL_ERROR_NUM is `0`

Choose one or more of these columns as the appropriate output for the extract_from_xml function.

i) Click *Finish*.

The software generates the function call in the current context and populates the output schema of the query with the output columns you specified.

With the data converted into the NRDM structure, you are ready to do appropriate transformation operations on it.

For example, if you want to load the NRDM structure to a target XML file, create an XML file target and connect the second query to it.

ℹ Note

If you find that you want to modify the function call, right-click the function call in the second query and choose *Modify Function Call*.

In this example, to extract XML data from a column of data type long, we created two queries: the first query to convert the data using the long_to_varchar function and the second query to add the extract_from_xml function.

Alternatively, you can use just one query by entering the function expression long_to_varchar directly into the first parameter of the function extract_from_xml. The first parameter of the function extract_from_xml can take a column of data type varchar or an expression that returns data of type varchar.

If the data type of the source column is not long but varchar, do not include the function long_to_varchar in your data flow.

2.10.5.1.2 Loading XML data into a column of the data type long

This scenario uses the load_to_xml function and the varchar_to_long function to convert an NRDM structure to scalar data of the varchar type in an XML format and load it to a column of the data type long.

In this example, you want to convert an NRDM structure for a purchase order to XML data using the function load_to_xml, and then load the data to an Oracle table column called *content*, which is of the long data type. Because the function load_to_xml returns a value of varchar data type, you use the function varchar_to_long to convert the value of varchar data type to a value of the data type long.

1. Create a query and connect a previous query or source (that has the NRDM structure of a purchase order) to it. In this query, create an output column of the data type varchar called *content*. Make sure the size of the column is big enough to hold the XML data.
2. From the *Mapping* area open the function wizard, click the category *Conversion Functions*, and then select the function *load_to_xml*.
3. Click *Next*.
4. Enter values for the input parameters.

   The function load_to_xml has seven parameters.
5. Click *Finish*.

   In the mapping area of the Query window, notice the function expression:

   ```
   load_to_xml(PO, 'PO', 1, '<?xml version="1.0" encoding = "UTF-8" ?>', NULL, 1,
   4000)
   ```

   In this example, this function converts the NRDM structure of purchase order PO to XML data and assigns the value to output column content.
6. Create another query with output columns matching the columns of the target table.
   a) Assume the column is called *content* and it is of the data type long.
   b) Open the function wizard from the mapping section of the query and select the *Conversion Functions* category
   c) Use the function varchar_to_long to map the input column *content* to the output column *content*.

The function varchar_to_long takes only one input parameter.

d) Enter a value for the input parameter.

```
varchar_to_long(content)
```

7. Connect this query to a database target.

Like the example using the extract_from_xml function, in this example, you used two queries. You used the first query to convert an NRDM structure to XML data and to assign the value to a column of varchar data type. You used the second query to convert the varchar data type to long.

You can use just one query if you use the two functions in one expression:

```
varchar_to_long( load_to_xml(PO, 'PO', 1, '<?xml version="1.0" encoding = "UTF-8" ?
>', NULL, 1, 4000) )
```

If the data type of the column in the target database table that stores the XML data is varchar, there is no need for varchar_to_long in the transformation.

Related Information

Reference Guide: Functions and Procedure [page 1511]

2.10.5.1.3 Extracting data quality XML strings using extract_from_xml function

This scenario uses the extract_from_xml function to extract XML data from the Geocoder, Global Suggestion Lists, Global Address Cleanse, and USA Regulatory Address Cleanse transforms.

The Geocoder transform, Global Suggestion Lists transform, and the suggestion list functionality in the Global Address Cleanse and USA Regulatory Address Cleanse transforms can output a field that contains an XML string. The transforms output the following fields that can contain XML.

Transform	XML output field	Output field description
Geocoder	Result_List	Contains an XML output string when multiple records are returned for a search. The content depends on the available data.
Global Address Cleanse Global Suggestion List USA Regulatory Address Cleanse	Suggestion_List	Contains an XML output string that includes all of the suggestion list component field values specified in the transform options.

Transform	XML output field	Output field description
		To output these fields as XML, you must choose XML as the output style in the transform options.

To use the data contained within the XML strings (for example, in a web application that uses the job published as a web service), you must extract the data. There are two methods that you can use to extract the data:

1. Insert a Query transform using the extract_from_xml function.
   With this method, you insert a Query transform into the data flow after the Geocoder, Global Suggestion Lists, Global Address Cleanse, or USA Regulatory Address Cleanse transform. Then you use the extract_from_xml function to parse the nested output data.
   This method is considered a best practice, because it provides parsed output data that is easily accessible to an integrator.
2. Develop a simple data flow that does not unnest the nested data.
   With this method, you simply output the output field that contains the XML string without unnesting the nested data.
   This method allows the application developer, or integrator, to dynamically select the output components in the final output schema before exposing it as a web service. The application developer must work closely with the data flow designer to understand the data flow behind a real-time web service. The application developer must understand the transform options and specify what to return from the return address suggestion list, and then unnest the XML output string to generate discrete address elements.

2.10.5.1.3.1 To extract data quality XML strings using extract_from_xml function

1. Create an XSD file for the output.
2. In the Format tab of the Local Object Library, create an XML Schema for your output XSD.
3. In the Format tab of the Local Object Library, create an XML Schema for the `gac_suggestion_list.xsd`, `global_suggestion_list.xsd`,`urac_suggestion_list.xsd`, or `result_list.xsd`.
4. In the data flow, include the following field in the Schema Out of the transform:
   - For the Global Address Cleanse, Global Suggestion Lists, and USA Regulatory Address Cleanse transforms, include the Suggestion_List field.
   - For the Geocoder transform, include the Result_List field
5. Add a Query transform after the Global Address Cleanse, Global Suggestion Lists,USA Regulatory Address Cleanse, or Geocoder transform. Complete it as follows.
6. Pass through all fields except the Suggestion_List or Result_List field from the Schema In to the Schema Out. To do this, drag fields directly from the input schema to the output schema.
7. In the Schema Out, right-click the Query node and select New Output Schema. Enter Suggestion_List or Result_List as the schema name (or whatever the field name is in your output XSD).
8. In the Schema Out, right-click the Suggestion_List or Result_List field and select Make Current.
9. In the Schema Out, right-click the Suggestion_List or Result_List list field and select New Function Call.
10. Select extract_from_xml from the Conversion Functions category and click Next. In the Define Input Parameter(s) window, enter the following information and click Next.

- ○ XML field name—Select the Suggestion_List or Result_List field from the upstream transform.
- ○ DTD or Schema name—Select the XML Schema that you created for the gac_suggestion_list.xsd, urac_suggestion_list.xsd, or result_list.xsd.
- ○ Enable validation—Enter 1 to enable validation.

11. Select LIST or RECORD from the left parameter list and click the right arrow button to add it to the Selected output parameters list.
12. Click Finish.
    The Schema Out includes the suggestion list/result list fields within the Suggestion_List or Result_List field.
13. Include the XML Schema for your output XML following the Query. Open the XML Schema to validate that the fields are the same in both the Schema In and the Schema Out.
14. If you are extracting data from a Global Address Cleanse, Global Suggestion Lists, or USA Regulatory Address Cleanse transform, and have chosen to output only a subset of the available suggestion list output fields in the Options tab, insert a second Query transform to specify the fields that you want to output. This allows you to select the output components in the final output schema before it is exposed as a web service.

2.11 Real-time Jobs

The software supports real-time data transformation. Real-time means that the software can receive requests from ERP systems and Web applications and send replies immediately after getting the requested data from a data cache or a second application. You define operations for processing on-demand messages by building real-time jobs in the Designer.

2.11.1 Request-response message processing

The message passed through a real-time system includes the information required to perform a business transaction. The content of the message can vary:

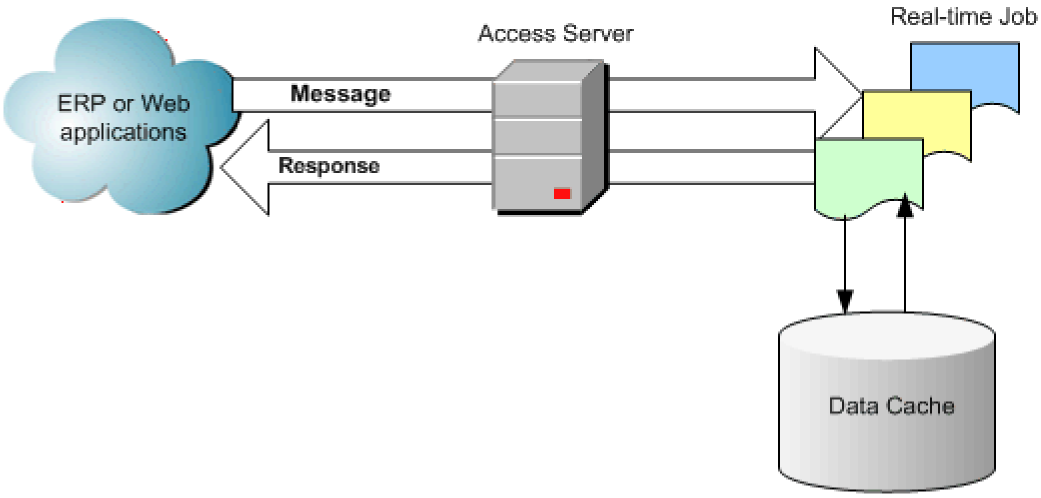
- It could be a sales order or an invoice processed by an ERP system destined for a data cache.
- It could be an order status request produced by a Web application that requires an answer from a data cache or back-office system.

The Access Server constantly listens for incoming messages. When a message is received, the Access Server routes the message to a waiting process that performs a predefined set of operations for the message type. The Access Server then receives a response for the message and replies to the originating application.

Two components support request-response message processing:

- *Access Server* — Listens for messages and routes each message based on message type.
- *Real-time job* — Performs a predefined set of operations for that message type and creates a response.

Processing might require that additional data be added to the message from a data cache or that the message data be loaded to a data cache. The Access Server returns the response to the originating application.



2.11.2 What is a real-time job?

The Designer allows you to define the processing of real-time messages using a real-time job. You create a different real-time job for each type of message your system can produce.

2.11.2.1 Real-time versus batch

Like a batch job, a real-time job extracts, transforms, and loads data. Real-time jobs "extract" data from the body of the message received and from any secondary sources used in the job. Each real-time job can extract data from a single message type. It can also extract data from other sources such as tables or files.

The same powerful transformations you can define in batch jobs are available in real-time jobs. However, you might use transforms differently in real-time jobs. For example, you might use branches and logic controls more often than you would in batch jobs. If a customer wants to know when they can pick up their order at your distribution center, you might want to create a CheckOrderStatus job using a look-up function to count order items and then a case transform to provide status in the form of strings: "No items are ready for pickup" or "X items in your order are ready for pickup" or "Your order is ready for pickup".

Also in real-time jobs, the software writes data to message targets and secondary targets in parallel. This ensures that each message receives a reply as soon as possible.

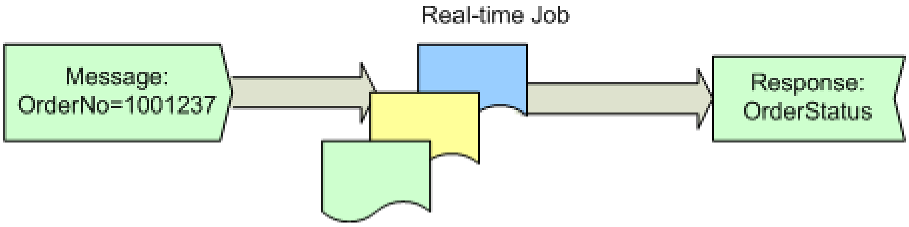
Unlike batch jobs, real-time jobs do not execute in response to a schedule or internal trigger; instead, real-time jobs execute as real-time services started through the Administrator. Real-time services then wait for messages from the Access Server. When the Access Server receives a message, it passes the message to a running real-time service designed to process this message type. The real-time service processes the message and returns a response. The real-time service continues to listen and process messages on demand until it receives an instruction to shut down.

2.11.2.2 Messages

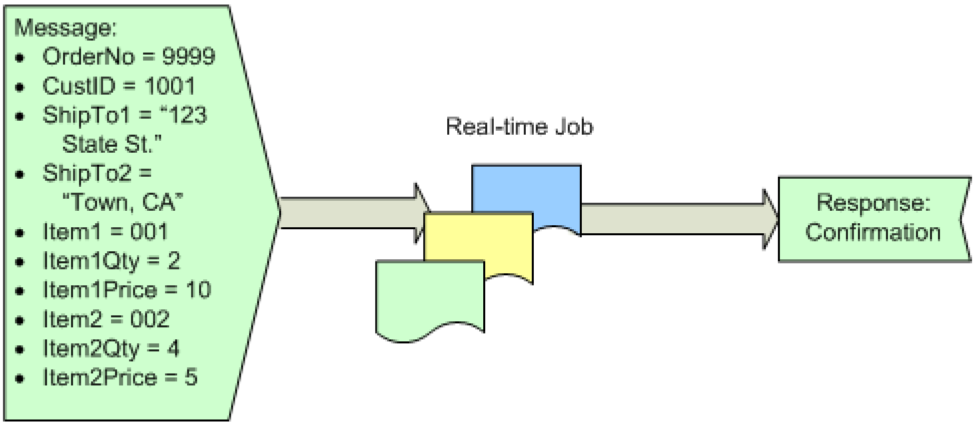
How you design a real-time job depends on what message you want it to process. Typical messages include information required to implement a particular business operation and to produce an appropriate response.

For example, suppose a message includes information required to determine order status for a particular order. The message contents might be as simple as the sales order number. The corresponding real-time job might use the input to query the right sources and return the appropriate product information.

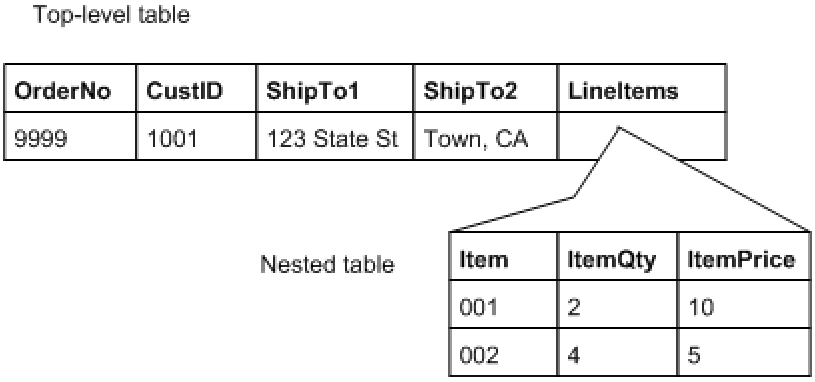
In this case, the message contains data that can be represented as a single column in a single-row table.



In a second case, a message could be a sales order to be entered into an ERP system. The message might include the order number, customer information, and the line-item details for the order. The message processing could return confirmation that the order was submitted successfully.



In this case, the message contains data that cannot be represented in a single table; the order header information can be represented by a table and the line items for the order can be represented by a second table. The software represents the header and line item data in the message in a nested relationship.


Top-level table

OrderNo	CustID	ShipTo1	ShipTo2	LineItems
9999	1001	123 State St	Town, CA	

Nested table

Item	ItemQty	ItemPrice
001	2	10
002	4	5

When processing the message, the real-time job processes all of the rows of the nested table for each row of the top-level table. In this sales order, both of the line items are processed for the single row of header information.

Real-time jobs can send only one row of data in a reply message (message target). However, you can structure message targets so that all data is contained in a single row by nesting tables within columns of a single, top-level table.

The software data flows support the nesting of tables within other tables.

Related Information

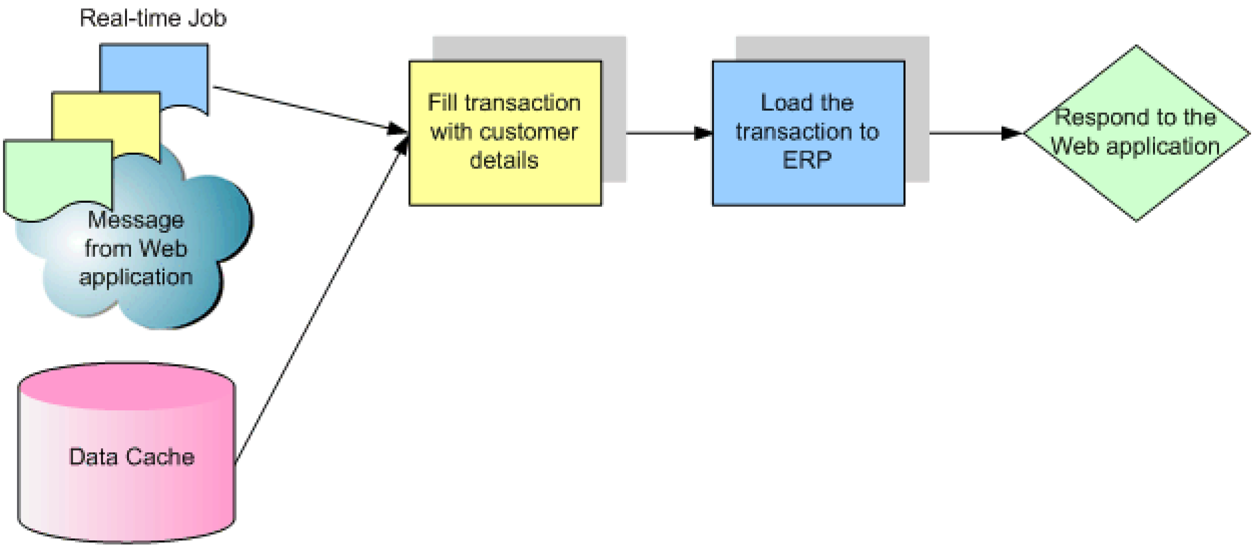
Nested Data [page 342]

2.11.2.3 Real-time job examples

These examples provide a high-level description of how real-time jobs address typical real-time scenarios. Later sections describe the actual objects that you would use to construct the logic in the Designer.

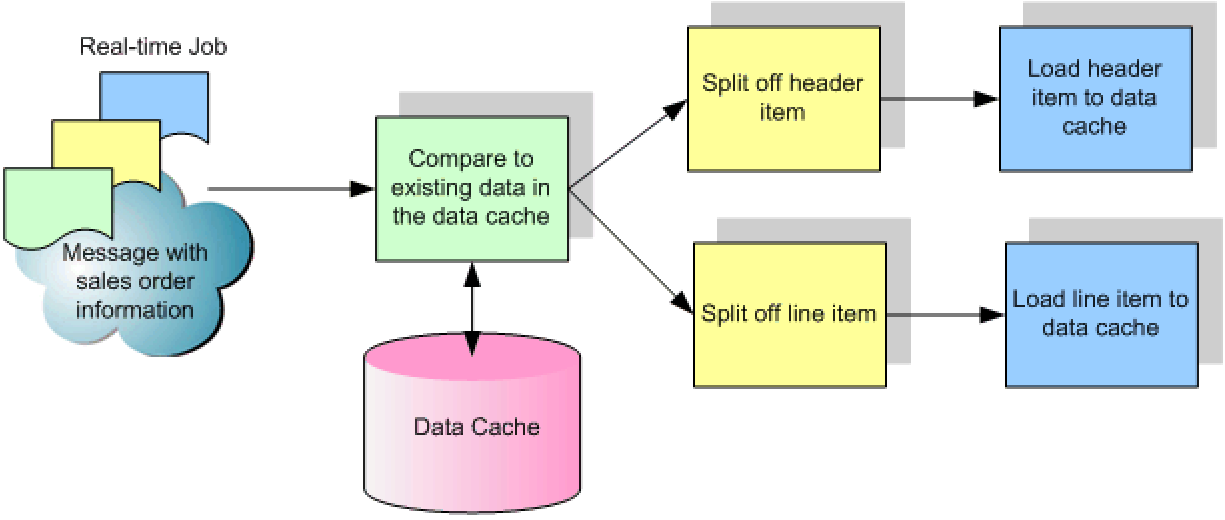
2.11.2.3.1 Loading transactions into a back-office application

A real-time job can receive a transaction from a Web application and load it to a back-office application (ERP, SCM, legacy). Using a query transform, you can include values from a data cache to supplement the transaction before applying it against the back-office application (such as an ERP system).


Real-time Job
Message from Web application
Data Cache
Fill transaction with customer details
Load the transaction to ERP
Respond to the Web application

2.11.2.3.2 Collecting back-office data into a data cache

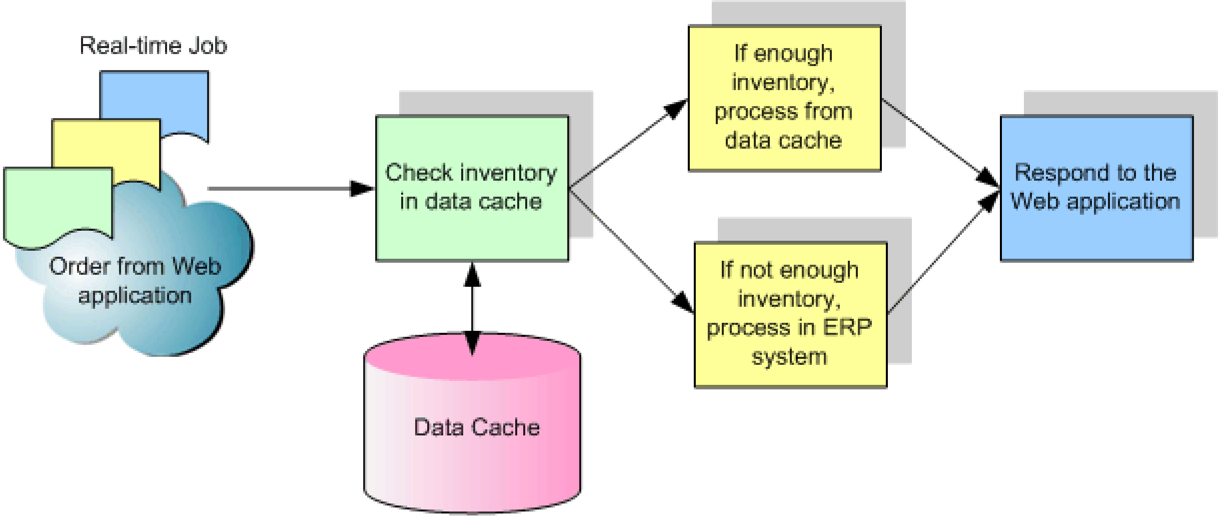
You can use messages to keep the data cache current. Real-time jobs can receive messages from a back-office application and load them into a data cache or data warehouse.



Real-time Job
Message with sales order information
Compare to existing data in the data cache
Data Cache
Split off header item
Load header item to data cache
Split off line item
Load line item to data cache

2.11.2.3.3 Retrieving values, data cache, back-office applications

You can create real-time jobs that use values from a data cache to determine whether or not to query the back-office application (such as an ERP system) directly.



Real-time Job
Order from Web application
Check inventory in data cache
Data Cache
If enough inventory, process from data cache
If not enough inventory, process in ERP system
Respond to the Web application

2.11.3 Creating real-time jobs

You can create real-time jobs using the same objects as batch jobs (data flows, work flows, conditionals, scripts, while loops, etc.). However, object usage must adhere to a valid real-time job model.

2.11.3.1 Real-time job models

In contrast to batch jobs, which typically move large amounts of data at scheduled times, a real-time job, once started as a real-time service, listens for a request. When a real-time job receives a request (typically to access small number of records), the software processes the request, returns a reply, and continues listening. This listen-process-listen logic forms a processing loop.

A real-time job is divided into three processing components: initialization, a real-time processing loop, and clean-up.

- The initialization component (optional) can be a script, work flow, data flow, or a combination of objects. It runs only when a real-time service starts.
- The real-time processing loop is a container for the job's single process logic. You can specify any number of work flows and data flows inside it.
- The clean-up component (optional) can be a script, work flow, data flow, or a combination of objects. It runs only when a real-time service is shut down.

In a real-time processing loop, a single message source must be included in the first step and a single message target must be included in the last step.

The following models support this rule:

- Single data flow model
- Multiple data flow model
- Request/Acknowledge data flow model

Related Information

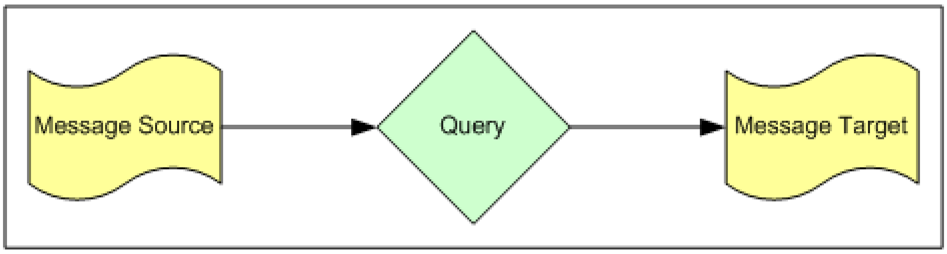
Single data flow model [page 377]

Multiple data flow model [page 378]

Supplement for SAP: SAP ERP and R/3 and Real-Time Jobs, IDoc sources in real-time jobs [page 2496]

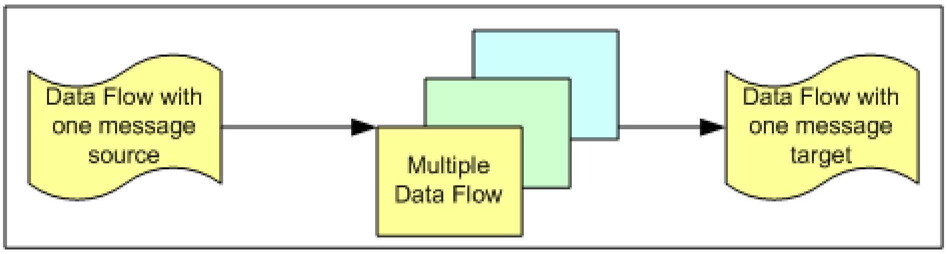
2.11.3.1.1 Single data flow model

With the single data flow model, you create a real-time job using a single data flow in its real-time processing loop. This single data flow must include a single message source and a single message target.



2.11.3.1.2 Multiple data flow model

The multiple data flow model allows you to create a real-time job using multiple data flows in its real-time processing loop.


Data Flow with one message source
Multiple Data Flow
Data Flow with one message target

By using multiple data flows, you can ensure that data in each message is completely processed in an initial data flow before processing for the next data flows starts. For example, if the data represents 40 items, all 40 must pass though the first data flow to a staging or memory table before passing to a second data flow. This allows you to control and collect all the data in a message at any point in a real-time job for design and troubleshooting purposes.

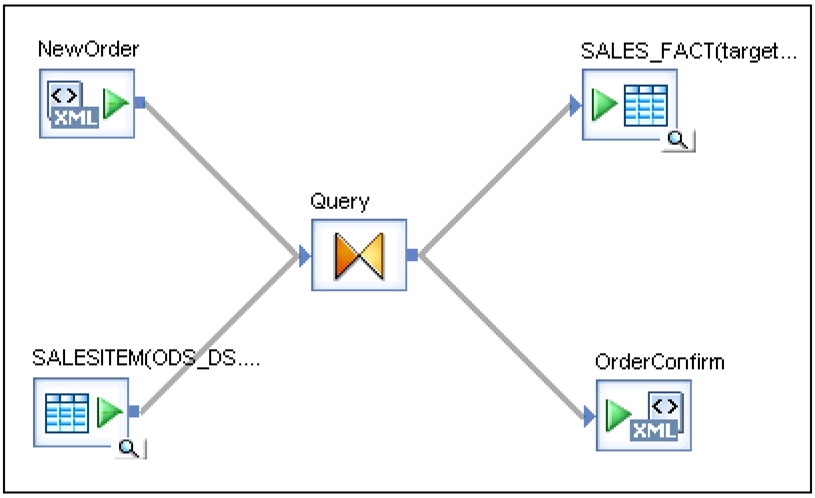
If you use multiple data flows in a real-time processing loop:

- The first object in the loop must be a data flow. This data flow must have one and only one message source.
- The last object in the loop must be a data flow. This data flow must have a message target.
- Additional data flows cannot have message sources or targets.
- You can add any number of additional data flows to the loop, and you can add them inside any number of work flows.
- All data flows can use input and/or output memory tables to pass data sets on to the next data flow. Memory tables store data in memory while a loop runs. They improve the performance of real-time jobs with multiple data flows.

2.11.3.2 Using real-time job models

2.11.3.2.1 Single data flow model

When you use a single data flow within a real-time processing loop your data flow diagram might look like this:



Notice that the data flow has one message source and one message target.

2.11.3.2.2 Multiple data flow model

When you use multiple data flows within a real-time processing loop your data flow diagrams might look like those in the following example scenario in which Data Services writes data to several targets according to your multiple data flow design.

Example scenario requirements:

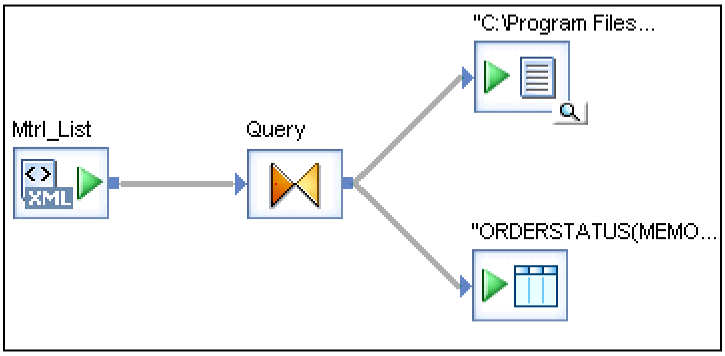
Your job must do the following tasks, completing each one before moving on to the next:

- Receive requests about the status of individual orders from a web portal and record each message to a backup flat file
- Perform a query join to find the status of the order and write to a customer database table.
- Reply to each message with the query join results

Solution:

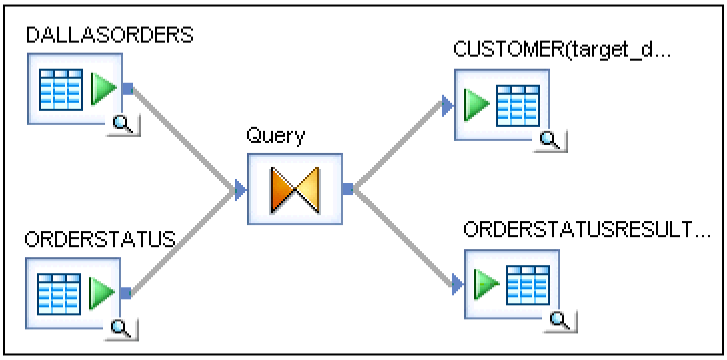
First, create a real-time job and add a data flow, a work flow, and another data flow to the real-time processing loop. Second, add a data flow to the work flow. Next, set up the tasks in each data flow:

- The first data flow receives the XML message (using an XML message source) and records the message to the flat file (flat file format target). Meanwhile, this same data flow writes the data into a memory table (table target).

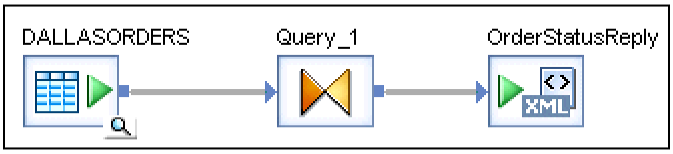


> Note
>
> You might want to create a memory table to move data to sequential data flows.

- The second data flow reads the message data from the memory table (table source), performs a join with stored data (table source), and writes the results to a database table (table target) and a new memory table (table target).
  Notice that this data flow has neither a message source nor a message target.



- The last data flow sends the reply. It reads the result of the join in the memory table (table source) and loads the reply (XML message target).



Related Information

Designing real-time applications [page 390]
Memory datastores [page 224]

2.11.3.3 To create a real-time job with a single data flow

1. In the Designer, create or open an existing project.

2. From the project area, right-click the white space and select *New Real-time job* from the shortcut menu.

   New_RTJob1 appears in the project area. The workspace displays the job's structure, which consists of two markers:

   ○ RT_Process_begins
   ○ Step_ends

   These markers represent the beginning and end of a real-time processing loop.
3. In the project area, rename *New_RTJob1*.

   Always add a prefix to job names with their job type. In this case, use the naming convention: RTJOB_JobName.

   Although saved real-time jobs are grouped together under the Job tab of the object library, job names may also appear in text editors used to create adapter or Web Services calls. In these cases, a prefix saved with the job name will help you identify it.
4. If you want to create a job with a single data flow:
   a) Click the data flow icon in the tool palette.

      You can add data flows to either a batch or real-time job. When you place a data flow icon into a job, you are telling Data Services to validate the data flow according the requirements of the job type (batch or real-time).
   b) Click inside the loop.

      The boundaries of a loop are indicated by begin and end markers. One message source and one message target are allowed in a real-time processing loop.
   c) Connect the begin and end markers to the data flow.
   d) Build the data flow including a message source and message target.
   e) Add, configure, and connect initialization object(s) and clean-up object(s) as needed.

2.11.4 Real-time source and target objects

Real-time jobs must contain a real-time source and/or target object. Those normally available are:

Object	Description	Used as a:	Software access
XML message	An XML message structured in a DTD or XML Schema format	Source or target	Directly or through adapters
Outbound message	A real-time message with an application-specific format (not readable by XML parser)	Target	Through an adapter

You can also use IDoc messages as real-time sources for SAP applications. For more information, see the *Supplement for SAP*.

Adding sources and targets to real-time jobs is similar to adding them to batch jobs, with the following additions:

For	Prerequisite	Object library location
XML messages	Import a DTD or XML Schema to define a format	*Formats* tab

For	Prerequisite	Object library location
Outbound message	Define an adapter datastore and import object metadata.	*Datastores* tab, under adapter datastore

Related Information

To import a DTD or XML Schema format [page 354]

Adapter datastores [page 233]

2.11.4.1 To view an XML message source or target schema

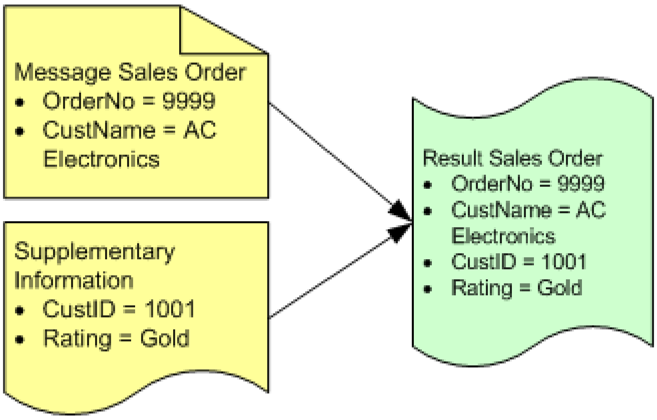
In the workspace of a real-time job, click the name of an XML message source or XML message target to open its editor.

If the XML message source or target contains nested data, the schema displays nested tables to represent the relationships among the data.

2.11.4.2 Secondary sources and targets

Real-time jobs can also have secondary sources or targets. For example, suppose you are processing a message that contains a sales order from a Web application. The order contains the customer name, but when you apply the order against your ERP system, you need to supply more detailed customer information.

Inside a data flow of a real-time job, you can supplement the message with the customer information to produce the complete document to send to the ERP system. The supplementary information might come from the ERP system itself or from a data cache containing the same information.



Tables and files (including XML files) as sources can provide this supplementary information.

The software reads data from secondary sources according to the way you design the data flow. The software loads data to secondary targets in parallel with a target message.

Add secondary sources and targets to data flows in real-time jobs as you would to data flows in batch jobs.

Related Information

Source and target objects [page 286]
Adding source or target objects to data flows [page 288]

2.11.4.3 Transactional loading of tables

Target tables in real-time jobs support transactional loading, in which the data resulting from the processing of a single data flow can be loaded into multiple tables as a single transaction. No part of the transaction applies if any part fails.

> Note
>
> Target tables in batch jobs also support transactional loading. However, use caution when you consider enabling this option for a batch job because it requires the use of memory, which can reduce performance when moving large amounts of data.

You can specify the order in which tables in the transaction are included using the target table editor. This feature supports a scenario in which you have a set of tables with foreign keys that depend on one with primary keys.

You can use transactional loading only when all the targets in a data flow are in the same datastore. If the data flow loads tables in more than one datastore, targets in each datastore load independently. While multiple targets in one datastore may be included in one transaction, the targets in another datastores must be included in another transaction.

You can specify the same transaction order or distinct transaction orders for all targets to be included in the same transaction. If you specify the same transaction order for all targets in the same datastore, the tables are still included in the same transaction but are loaded together. Loading is committed after all tables in the transaction finish loading.

If you specify distinct transaction orders for all targets in the same datastore, the transaction orders indicate the loading orders of the tables. The table with the smallest transaction order is loaded first, and so on, until the table with the largest transaction order is loaded last. No two tables are loaded at the same time. Loading is committed when the last table finishes loading.

2.11.4.4 Design tips for data flows in real-time jobs

Keep in mind the following when you are designing data flows:

- If you include a table in a join with a real-time source, the software includes the data set from the real-time source as the outer loop of the join. If more than one supplementary source is included in the join, you can control which table is included in the next outer-most loop of the join using the join ranks for the tables.

- In real-time jobs, do not cache data from secondary sources unless the data is static. The data will be read when the real-time job starts and will not be updated while the job is running.
- If no rows are passed to the XML target, the real-time job returns an empty response to the Access Server. For example, if a request comes in for a product number that does not exist, your job might be designed in such a way that no data passes to the reply message. You might want to provide appropriate instructions to your user (exception handling in your job) to account for this type of scenario.
- If more than one row passes to the XML target, the target reads the first row and discards the other rows. To avoid this issue, use your knowledge of the software's Nested Relational Data Model (NRDM) and structure your message source and target formats so that one "row" equals one message. With NRDM, you can structure any amount of data into a single "row" because columns in tables can contain other tables.
- Recovery mechanisms are not supported in real-time jobs.

Related Information

Reference Guide: Objects, Real-time job [page 942]

Nested Data [page 342]

2.11.5 Testing real-time jobs

2.11.5.1 Executing a real-time job in test mode

You can test real-time job designs without configuring the job as a service associated with an Access Server. In test mode, you can execute a real-time job using a sample source message from a file to determine if the software produces the expected target message.

2.11.5.1.1 To specify a sample XML message and target test file

1. In the XML message source and target editors, enter a file name in the *XML test file* box.

   Enter the full path name for the source file that contains your sample data. Use paths for both test files relative to the computer that runs the Job Server for the current repository.
2. Execute the job.

   Test mode is always enabled for real-time jobs. The software reads data from the source test file and loads it into the target test file.

2.11.5.2 Using View Data

To ensure that your design returns the results you expect, execute your job using View Data. With View Data, you can capture a sample of your output data to ensure that your design is working.

Related Information

Design and Debug [page 685]

2.11.5.3 Using an XML file target

You can use an "XML file target" to capture the message produced by a data flow while allowing the message to be returned to the Access Server.

Just like an XML message, you define an XML file by importing a DTD or XML Schema for the file, then dragging the format into the data flow definition. Unlike XML messages, you can include XML files as sources or targets in batch and real-time jobs.

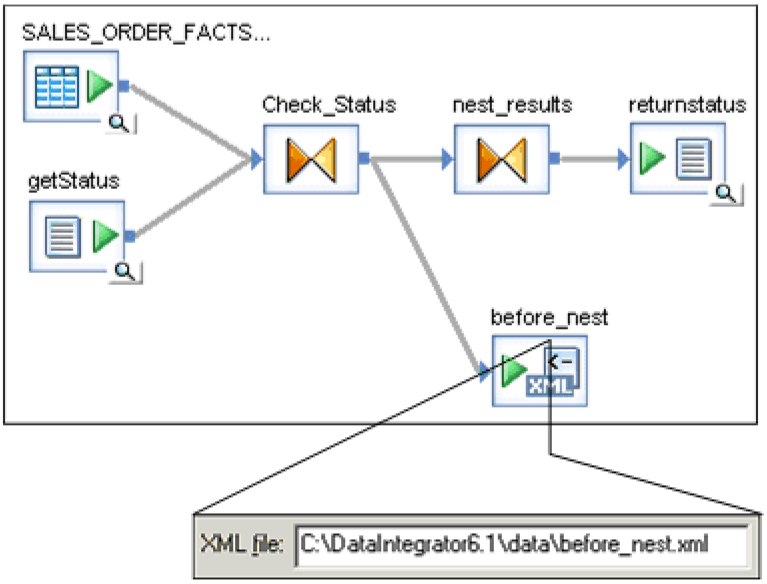
2.11.5.3.1 To use a file to capture output from steps in a real-time job

1. In the *Formats* tab of the object library, drag the DTD or XML Schema into a data flow of a real-time job.

   A menu prompts you for the function of the file.
2. Choose *Make XML File Target*.

   The XML file target appears in the workspace.
3. In the file editor, specify the location to which the software writes data.

   Enter a file name relative to the computer running the Job Server.
4. Connect the output of the step in the data flow that you want to capture to the input of the file.



XML file: C:\DataIntegrator6.1\data\before_nest.xml

2.11.6 Building blocks for real-time jobs

2.11.6.1 Supplementing message data

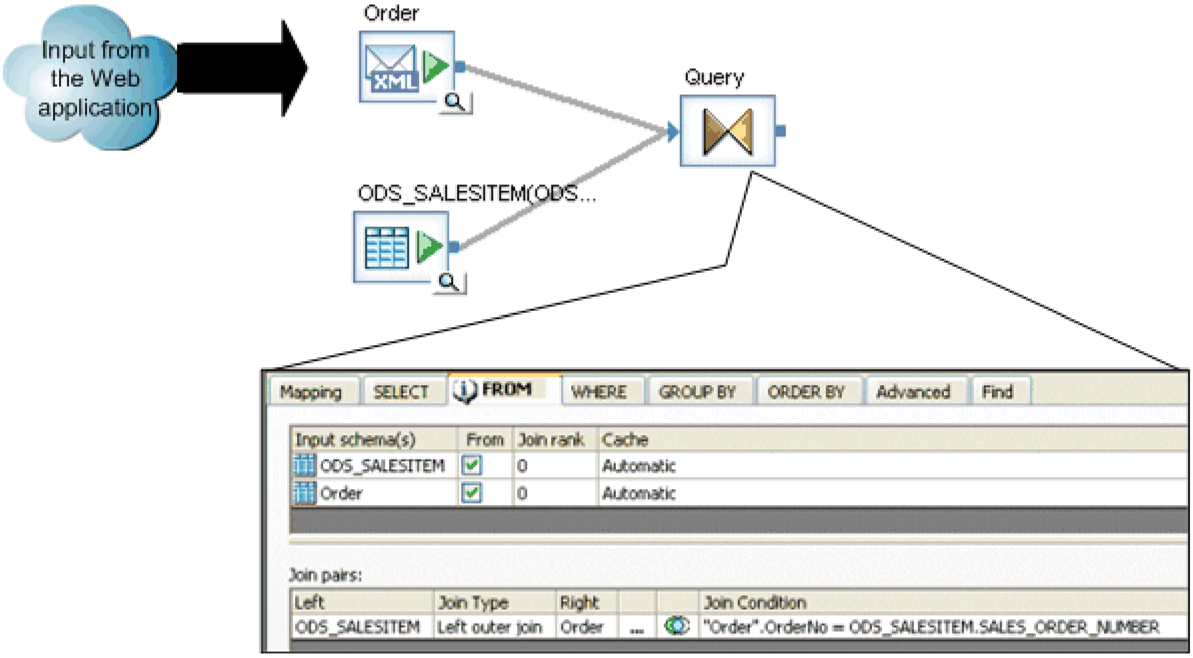
The data included in messages from real-time sources might not map exactly to your requirements for processing or storing the information. If not, you can define steps in the real-time job to supplement the message information.

One technique for supplementing the data in a real-time source includes these steps:

1. Include a table or file as a source.

   In addition to the real-time source, include the files or tables from which you require supplementary information.
2. Use a query to extract the necessary data from the table or file.
3. Use the data in the real-time source to find the necessary supplementary data.

   You can include a join expression in the query to extract the specific values required from the supplementary source.



Input schema(s)	From	Join rank	Cache
ODS_SALESITEM	☑	0	Automatic
Order	☑	0	Automatic

Join pairs:

Left	Join Type	Right			Join Condition
ODS_SALESITEM	Left outer join	Order	...		"Order".OrderNo = ODS_SALESITEM.SALES_ORDER_NUMBER

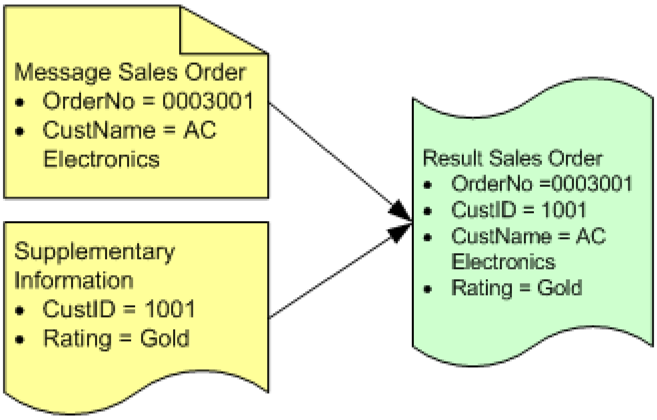
The Join Condition joins the two input schemas resulting in output for only the sales item document and line items included in the input from the application.

Be careful to use data in the join that is guaranteed to return a value. If no value returns from the join, the query produces no rows and the message returns to the Access Server empty. If you cannot guarantee that a value returns, consider these alternatives:

- *Lookup function call*—Returns a default value if no match is found.
- *Outer join*—Always returns a value, even if no match is found.

2.11.6.1.1 To supplement message data

In this example, a request message includes sales order information and its reply message returns order status. The business logic uses the customer number and priority rating to determine the level of status to return. The message includes only the customer name and the order number. A real-time job is then defined to retrieve the customer number and rating from other sources before determining the order status.



1. Include the real-time source in the real-time job.
2. Include the supplementary source in the real-time job.

   This source could be a table or file. In this example, the supplementary information required doesn't change very often, so it is reasonable to extract the data from a data cache rather than going to an ERP system directly.
3. Join the sources.

   In a query transform, construct a join on the customer name:

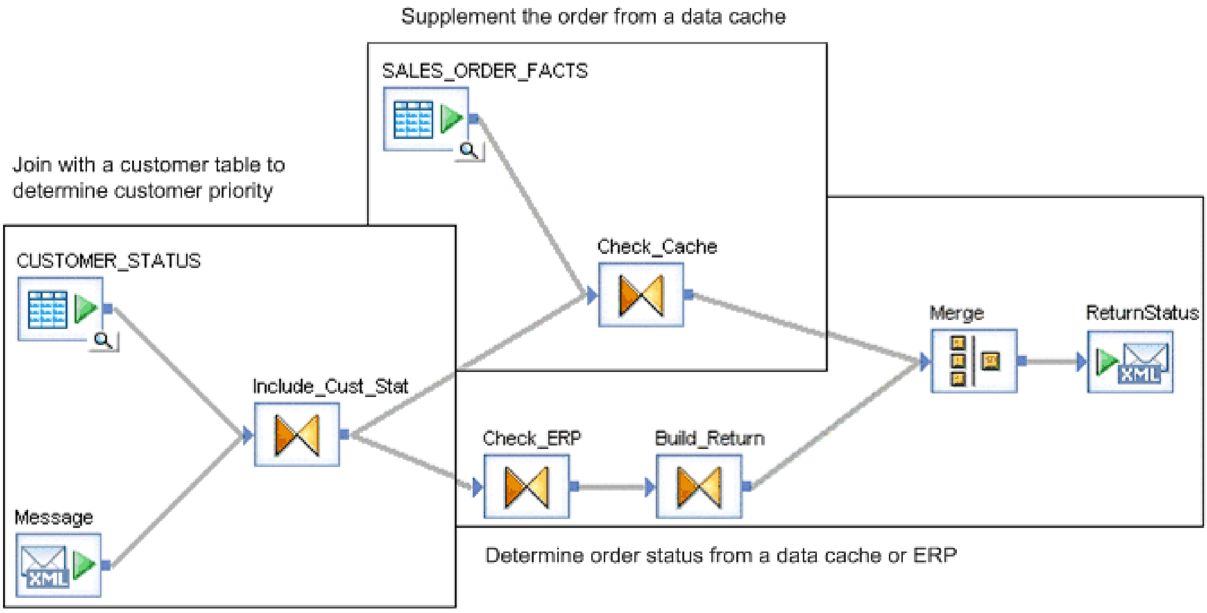
   ```
   Message.CustName = Cust_Status.CustName
   ```

   You can construct the output to include only the columns that the real-time job needs to determine order status.
4. Complete the real-time job to determine order status.

   The example shown here determines order status in one of two methods based on the customer status value. Order status for the highest ranked customers is determined directly from the ERP. Order status for other customers is determined from a data cache of sales order information.

   The logic can be arranged in a single or multiple data flows. The illustration below shows a single data flow model.

   Both branches return order status for each line item in the order. The data flow merges the results and constructs the response. The next section describes how to design branch paths in a data flow.



Supplement the order from a data cache
SALES_ORDER_FACTS
Join with a customer table to determine customer priority
Check_Cache
CUSTOMER_STATUS
Merge
ReturnStatus
XML
Include_Cust_Stat
Check_ERP
Build_Return
Message
XML
Determine order status from a data cache or ERP

2.11.6.2 Branching data flow based on a data cache value

One of the most powerful things you can do with a real-time job is to design logic that determines whether responses should be generated from a data cache or if they must be generated from data in a back-office application (ERP, SCM, CRM).

Here is one technique for constructing this logic:

1. Determine the rule for when to access the data cache and when to access the back-office application.
2. Compare data from the real-time source with the rule.
3. Define each path that could result from the outcome.

   You might need to consider the case where the rule indicates back-office application access, but the system is not currently available.
4. Merge the results from each path into a single data set.
5. Route the single result to the real-time target.

   You might need to consider error-checking and exception-handling to make sure that a value passes to the target. If the target receives an empty set, the real-time job returns an empty response (begin and end XML tags only) to the Access Server.

This example describes a section of a real-time job that processes a new sales order. The section is responsible for checking the inventory available of the ordered products—it answers the question, "is there enough inventory on hand to fill this order?"

The rule controlling access to the back-office application indicates that the inventory (Inv) must be more than a pre-determined value (IMargin) greater than the ordered quantity (Qty) to consider the data cached inventory value acceptable.

The software makes a comparison for each line item in the order they are mapped.

Table 6: Incoming sales order

OrderNo	CustID	LineItem		
		Item	Material	Qty
9999	1001	001	7333	300
		002	2288	1400

Table 7: Inventory data cache

Material	Inv	IMargin
7333	600	100
2288	1500	200

Note

The quantity of items in the sales order is compared to inventory values in the data cache.

2.11.6.3 Calling application functions

A real-time job can use application functions to operate on data. You can include tables as input or output parameters to the function.

Application functions require input values for some parameters and some can be left unspecified. You must determine the requirements of the function to prepare the appropriate inputs.

To make up the input, you can specify the top-level table, top-level columns, and any tables nested one-level down relative to the tables listed in the FROM clause of the context calling the function. If the application function includes a structure as an input parameter, you must specify the individual columns that make up the structure.

A data flow may contain several steps that call a function, retrieve results, then shape the results into the columns and tables required for a response.

2.11.7 Designing real-time applications

The software provides a reliable and low-impact connection between a Web application and back-office applications such as an enterprise resource planning (ERP) system. Because each implementation of an ERP system is different and because the software includes versatile decision support logic, you have many opportunities to design a system that meets your internal and external information and resource needs.

2.11.7.1 Reducing queries requiring back-office application access

This section provides a collection of recommendations and considerations that can help reduce the time you spend experimenting in your development cycles.

The information you allow your customers to access through your Web application can impact the performance that your customers see on the Web. You can maximize performance through your Web application design decisions. In particular, you can structure your application to reduce the number of queries that require direct back-office (ERP, SCM, Legacy) application access.

For example, if your ERP system supports a complicated pricing structure that includes dependencies such as customer priority, product availability, or order quantity, you might not be able to depend on values from a data cache for pricing information. The alternative might be to request pricing information directly from the ERP system. ERP system access is likely to be much slower than direct database access, reducing the performance your customer experiences with your Web application.

To reduce the impact of queries requiring direct ERP system access, modify your Web application. Using the pricing example, design the application to avoid displaying price information along with standard product information and instead show pricing only after the customer has chosen a specific product and quantity. These techniques are evident in the way airline reservations systems provide pricing information—a quote for a specific flight—contrasted with other retail Web sites that show pricing for every item displayed as part of product catalogs.

2.11.7.2 Messages from real-time jobs to adapter instances

If a real-time job will send a message to an adapter instance, refer to the adapter documentation to decide if you need to create a message function call or an outbound message.

- Message function calls allow the adapter instance to collect requests and send replies.
- Outbound message objects can only send outbound messages. They cannot be used to receive messages.

Related Information

Importing metadata through an adapter datastore [page 236]

2.11.7.3 Real-time service invoked by an adapter instance

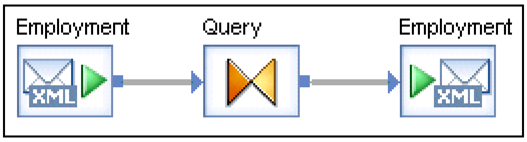
This section uses terms consistent with Java programming. (Please see your adapter SDK documentation for more information about terms such as operation instance and information resource.)

When an operation instance (in an adapter) gets a message from an information resource, it translates it to XML (if necessary), then sends the XML message to a real-time service.

In the real-time service, the message from the adapter is represented by a DTD or XML Schema object (stored in the *Formats* tab of the object library). The DTD or XML Schema represents the data schema for the information resource.

The real-time service processes the message from the information resource (relayed by the adapter) and returns a response.

In the example data flow below, the Query processes a message (here represented by "Employment") received from a source (an adapter instance), and returns the response to a target (again, an adapter instance).



2.12 Embedded Data Flows

The software provides an easy-to-use option to create embedded data flows.

2.12.1 Overview of embedded data flows

An embedded data flow is a data flow that is called from inside another data flow. Data passes into or out of the embedded data flow from the parent flow through a single source or target. The embedded data flow can contain any number of sources or targets, but only one input or one output can pass data to or from the parent data flow.

You can create the following types of embedded data flows:

Type	Use when you want to...
One input	Add an embedded data flow at the end of a data flow.
One output	Add an embedded data flow at the beginning of a data flow.
No input or output	Replicate an existing data flow.

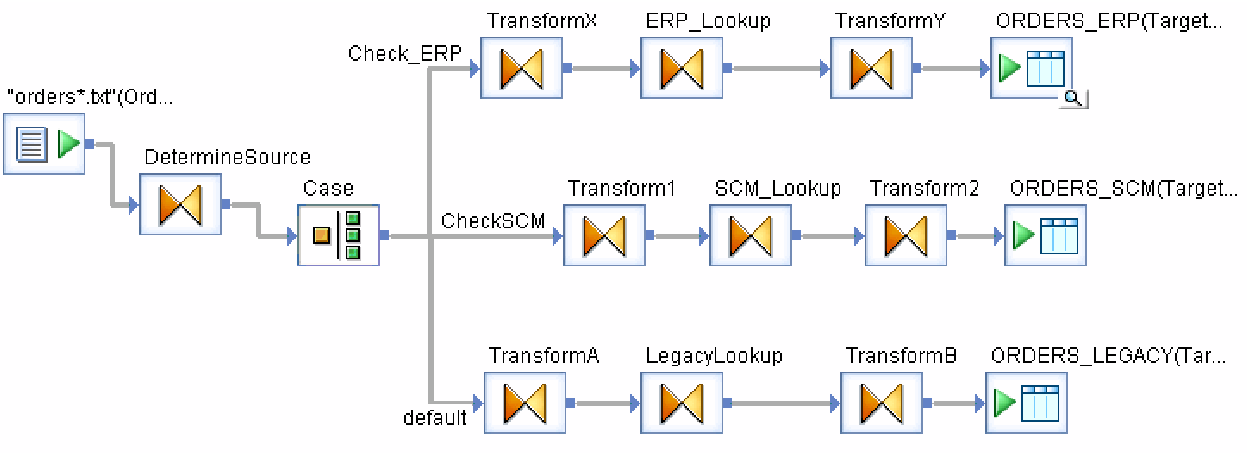
An embedded data flow is a design aid that has no effect on job execution. When the software executes the parent data flow, it expands any embedded data flows, optimizes the parent data flow, and then executes it.

Use embedded data flows to:

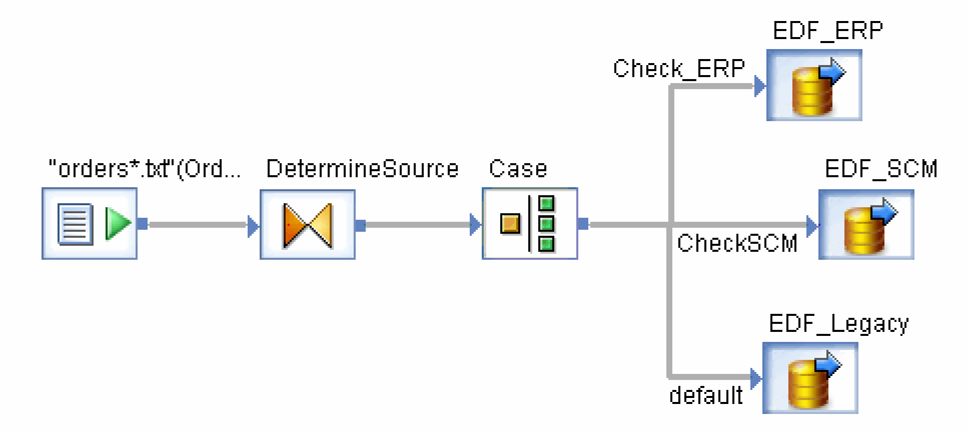
- Simplify data flow display. Group sections of a data flow in embedded data flows to allow clearer layout and documentation.
- Reuse data flow logic. Save logical sections of a data flow so you can use the exact logic in other data flows, or provide an easy way to replicate the logic and modify it for other flows.
- Debug data flow logic. Replicate sections of a data flow as embedded data flows so you can execute them independently.

2.12.2 Example of when to use embedded data flows

In this example, a data flow uses a single source to load three different target systems. The Case transform sends each row from the source to different transforms that process it to get a unique target output.



You can simplify the parent data flow by using embedded data flows for the three different cases.



2.12.3 Creating embedded data flows

There are two ways to create embedded data flows.

- Select objects within a data flow, right-click, and select *Make Embedded Data Flow*.
- Drag a complete and fully validated data flow from the object library into an open data flow in the workspace. Then:
  - Open the data flow you just added.
  - Right-click one object you want to use as an input or as an output port and select *Make Port* for that object.
    The software marks the object you select as the connection point for this embedded data flow.

> **Note**
>
> You can specify only one port, which means that the embedded data flow can appear only at the beginning or at the end of the parent data flow.
>
> Data Services ignores some physical files that are required for sources and targets with assigned ports.
>
> - When you use an embedded data flow, data will flow directly from the caller to the transform(s) next to the port source.
> - When you use a data flow directly, Data Services uses the physical files in sources and targets, but ignores the ports.

2.12.3.1 Using the Make Embedded Data Flow option

2.12.3.1.1 To create an embedded data flow

1. Select objects from an open data flow using one of the following methods:
   - Click the white space and drag the rectangle around the objects.
   - CTRL-click each object.

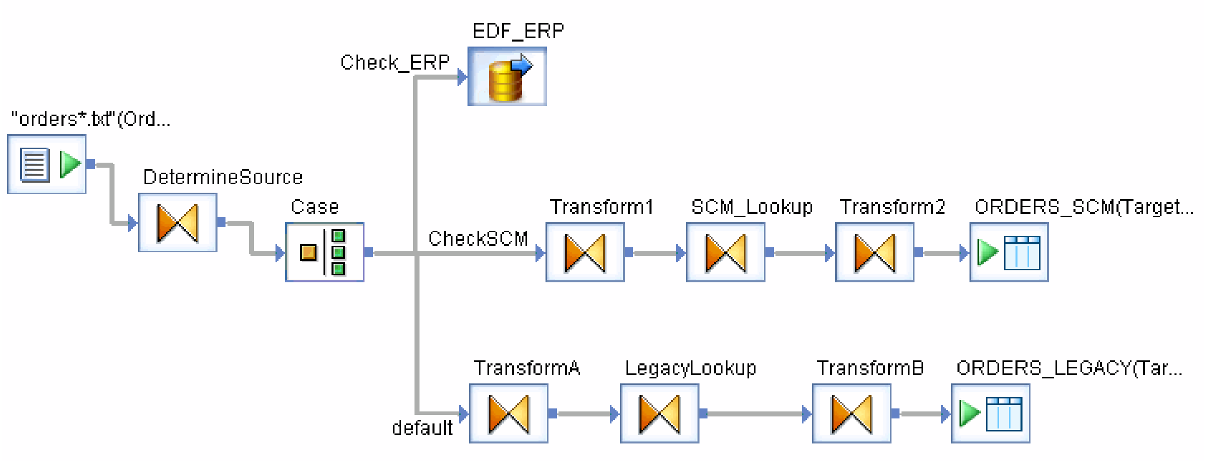
   Ensure that the set of objects you select are:
   - All connected to each other.
   - Connected to other objects according to the type of embedded data flow you want to create such as one input, one output, or no input or output.
2. Right-click and select *Make Embedded Data Flow*.

   The Create Embedded Data Flow window opens, with the embedded data flow connected to the parent by one input object.
3. Name the embedded data flow using the convention `EDF_` **`<EDFName>`** for example `EDF_ERP`.

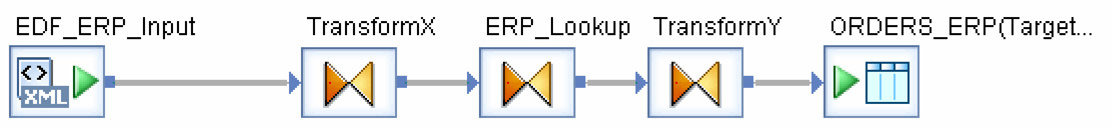
   If you deselect the Replace objects in original data flow box, the software will not make a change in the original data flow. The software saves the new embedded data flow object to the repository and displays it in the object library under the Data Flows tab.

   You can use an embedded data flow created without replacement as a stand-alone data flow for troubleshooting.

   If Replace objects in original data flow is selected, the original data flow becomes a parent data flow, which has a call to the new embedded data flow.
4. Click *OK*.



   The embedded data flow appears in the new parent data flow.
5. Click the name of the embedded data flow to open it.



6. Notice that the software created a new object, `EDF_ERP_Input`, which is the input port that connects this embedded data flow to the parent data flow.

When you use the *Make Embedded Data flow* option, the software automatically creates an input or output object based on the object that is connected to the embedded data flow when it is created.

For example, if an embedded data flow has an output connection, the embedded data flow will include a target XML file object labeled **`<EDFName>`**`_Output`.

The naming conventions for each embedded data flow type are:

Type	Naming conventions
One input	**`<EDFName>`**`_Input`
One output	**`<EDFName>`**`_Output`
No input or output	The software creates an embedded data flow without an input or output object.

2.12.3.2 Creating embedded data flows from existing flows

To call an existing data flow from inside another data flow, put the data flow inside the parent data flow, then mark which source or target to use to pass data between the parent and the embedded data flows.

2.12.3.2.1 To create an embedded data flow out of an existing data flow

1. Drag an existing valid data flow from the object library into a data flow that is open in the workspace.
2. Consider renaming the flow using the `EDF_`**`<EDFName>`** naming convention.

   The embedded data flow appears without any arrowheads (ports) in the workspace.
3. Open the embedded data flow.
4. Right-click a source or target object (file or table) and select *Make Port*.

   > i Note
   >
   > Ensure that you specify only one input or output port.

Like a normal data flow, different types of embedded data flow ports are indicated by directional markings on the embedded data flow icon.

2.12.3.3 Using embedded data flows

When you create and configure an embedded data flow using the *Make Embedded Data Flow* option, the software creates new input or output XML file and saves the schema in the repository as an XML Schema. You can reuse an embedded data flow by dragging it from the *Data Flow* tab of the object library into other data flows. To save mapping time, you might want to use the Update Schema option or the Match Schema option.

The following example scenario uses both options:

- Create data flow 1.
- Select objects in data flow 1, and create embedded data flow 1 so that parent data flow 1 calls embedded data flow 1.
- Create data flow 2 and data flow 3 and add embedded data flow 1 to both of them.
- Go back to data flow 1. Change the schema of the object preceding embedded data flow 1 and use the Update Schema option with embedded data flow 1. It updates the schema of embedded data flow 1 in the repository.
- Now the schemas in data flow 2 and data flow 3 that are feeding into embedded data flow 1 will be different from the schema the embedded data flow expects.
- Use the Match Schema option for embedded data flow 1 in both data flow 2 and data flow 3 to resolve the mismatches at runtime. The Match Schema option only affects settings in the current data flow.

The following sections describe the use of the Update Schema and Match Schema options in more detail.

2.12.3.3.1 Updating Schemas

The software provides an option to update an input schema of an embedded data flow. This option updates the schema of an embedded data flow's input object with the schema of the preceding object in the parent data flow. All occurrences of the embedded data flow update when you use this option.

2.12.3.3.1.1 To update a schema

1. Open the embedded data flow's parent data flow.
2. Right-click the embedded data flow object and select *Update Schema*.

2.12.3.3.2 Matching data between parent and embedded data flow

The schema of an embedded data flow's input object can match the schema of the preceding object in the parent data flow by name or position. A match by position is the default.

2.12.3.3.2.1 To specify how schemas should be matched

1. Open the embedded data flow's parent data flow.
2. Right-click the embedded data flow object and select *Match SchemaBy Name* or *Match SchemaBy Position*.

   The Match Schema option only affects settings for the current data flow.

Data Services also allows the schema of the preceding object in the parent data flow to have more or fewer columns than the embedded data flow. The embedded data flow ignores additional columns and reads missing columns as NULL.

Columns in both schemas must have identical or convertible data types. See the section on "Type conversion" in the *Reference Guide* for more information.

2.12.3.3.3 Deleting embedded data flow objects

You can delete embedded data flow ports, or remove entire embedded data flows.

2.12.3.3.3.1 To remove a port

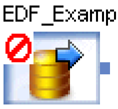
Right-click the input or output object within the embedded data flow and deselect *Make Port*. Data Services removes the connection to the parent object.

> Note
>
> You cannot remove a port simply by deleting the connection in the parent flow.

2.12.3.3.3.2 To remove an embedded data flow

Select it from the open parent data flow and choose *Delete* from the right-click menu or edit menu.

If you delete embedded data flows from the object library, the embedded data flow icon appears with a red circle-slash flag in the parent data flow.



Delete these defunct embedded data flow objects from the parent data flows.

2.12.3.4 Separately testing an embedded data flow

Embedded data flows can be tested by running them separately as regular data flows.

1. Specify an XML file for the input port or output port.

   When you use the Make Embedded Data Flow option, an input or output XML file object is created and then (optional) connected to the preceding or succeeding object in the parent data flow. To test the XML file without a parent data flow, click the name of the XML file to open its source or target editor to specify a file name.

2. Put the embedded data flow into a job.
3. Run the job.

You can also use the following features to test embedded data flows:

- View Data to sample data passed into an embedded data flow.
- Auditing statistics about the data read from sources, transformed, and loaded into targets, and rules about the audit statistics to verify the expected data is processed.

Related Information

Reference Guide: XML file [page 1004]
Design and Debug [page 685]

2.12.3.5 Troubleshooting embedded data flows

The following situations produce errors:

- Both an input port and output port are specified in an embedded data flow.
- Trapped defunct data flows.
- Deleted connection to the parent data flow while the *Make Port* option, in the embedded data flow, remains selected.
- Transforms with splitters (such as the Case transform) specified as the output port object because a splitter produces multiple outputs, and embedded data flows can only have one.
- Variables and parameters declared in the embedded data flow that are not also declared in the parent data flow.
- Embedding the same data flow at any level within itself.
  You can however have unlimited embedding levels. For example, DF1 data flow calls EDF1 embedded data flow which calls EDF2.

Related Information

To remove an embedded data flow [page 397]
To remove a port [page 397]

2.13 Variables and Parameters

This section contains information about the following:

- Adding and defining local and global variables for jobs

- Using environment variables
- Using substitution parameters and configurations

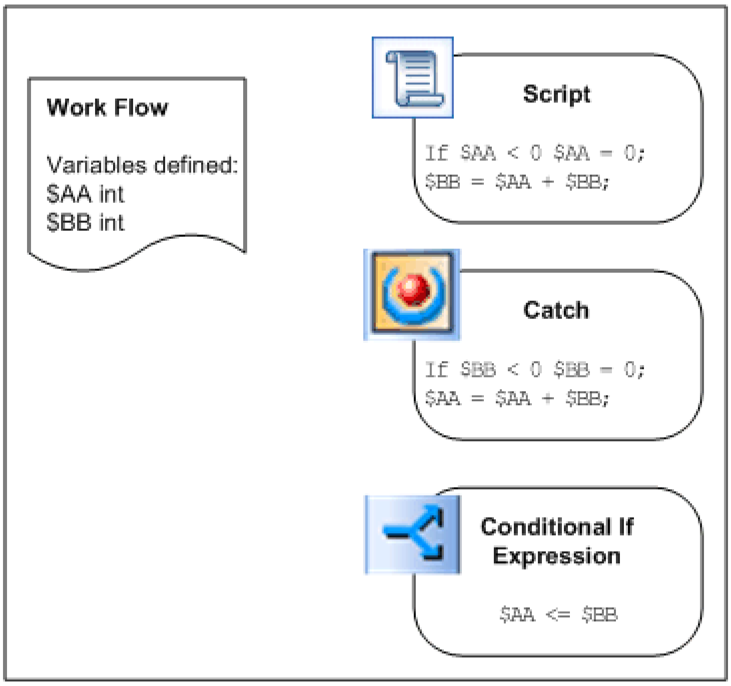
2.13.1 Overview of variables and parameters

You can increase the flexibility and reusability of work flows and data flows by using local and global variables when you design your jobs. Variables are symbolic placeholders for values. The data type of a variable can be any supported by the software such as an integer, decimal, date, or text string.

You can use variables in expressions to facilitate decision-making or data manipulation (using arithmetic or character substitution). For example, a variable can be used in a `LOOP` or `IF` statement to check a variable's value to decide which step to perform:

```
If $<amount_owed> > 0 print('$<invoice.doc>');
```

If you define variables in a job or work flow, the software typically uses them in a script, catch, or conditional process.



```
If $AA < 0 $AA = 0;
$BB = $AA + $BB;
```

```
If $BB < 0 $BB = 0;
$AA = $AA + $BB;
```

```
$AA <= $BB
```

You can use variables inside data flows. For example, use them in a custom function or in the `WHERE` clause of a query transform.

In the software, local variables are restricted to the object in which they are created (job or work flow). You must use parameters to pass local variables to child objects (work flows and data flows).

Global variables are restricted to the job in which they are created; however, they do not require parameters to be passed to work flows and data flows.

ℹ Note

If you have work flows that are running in parallel, the global variables are not assigned.

Parameters are expressions that pass to a work flow or data flow when they are called in a job.

You create local variables, parameters, and global variables using the Variables and Parameters window in the Designer.

You can set values for local or global variables in script objects. You can also set global variable values using external job, execution, or schedule properties.

Using global variables provides you with maximum flexibility. For example, during production you can change values for default global variables at runtime from a job's schedule or *SOAP* call without having to open a job in the Designer.

Variables can be used as file names for:

- Flat file sources and targets
- XML file sources and targets
- XML message targets (executed in the Designer in test mode)
- IDoc file sources and targets (in an SAP application environment)
- IDoc message sources and targets (SAP application environment)

Related Information

Management Console Guide: Administrator, Support for Web Services [page 1945]

2.13.2 The Variables and Parameters window

The software displays the variables and parameters defined for an object in the *Variables and Parameters* window.

2.13.2.1 To view the variables and parameters in each job, work flow, or data flow

1. In the *Tools* menu, select *Variables*.

   The *Variables and Parameters* window opens.
2. From the object library, double-click an object, or from the project area click an object to open it in the workspace.

   The Context box in the window changes to show the object you are viewing. If there is no object selected, the window does not indicate a context.

The Variables and Parameters window contains two tabs.

The *Definitions* tab allows you to create and view variables (name and data type) and parameters (name, data type, and parameter type) for an object type. Local variable and parameters can only be set at the work flow and data flow level. Global variables can only be set at the job level.

The following table lists what type of variables and parameters you can create using the Variables and Parameters window when you select different objects.

Object type	What you can create for the object	Used by
Job	Local variables Global variables	A script or conditional in the job. Any object in the job.
Work flow	Local variables Parameters	This work flow or passed down to other work flows or data flows using a parameter. Parent objects to pass local variables. Work flows may also return variables or parameters to parent objects.
Data flow	Parameters	A WHERE clause, column mapping, or a function in the data flow. Data flows cannot return output values.

The *Calls* tab allows you to view the name of each parameter defined for all objects in a parent object's definition. You can also enter values for each parameter.

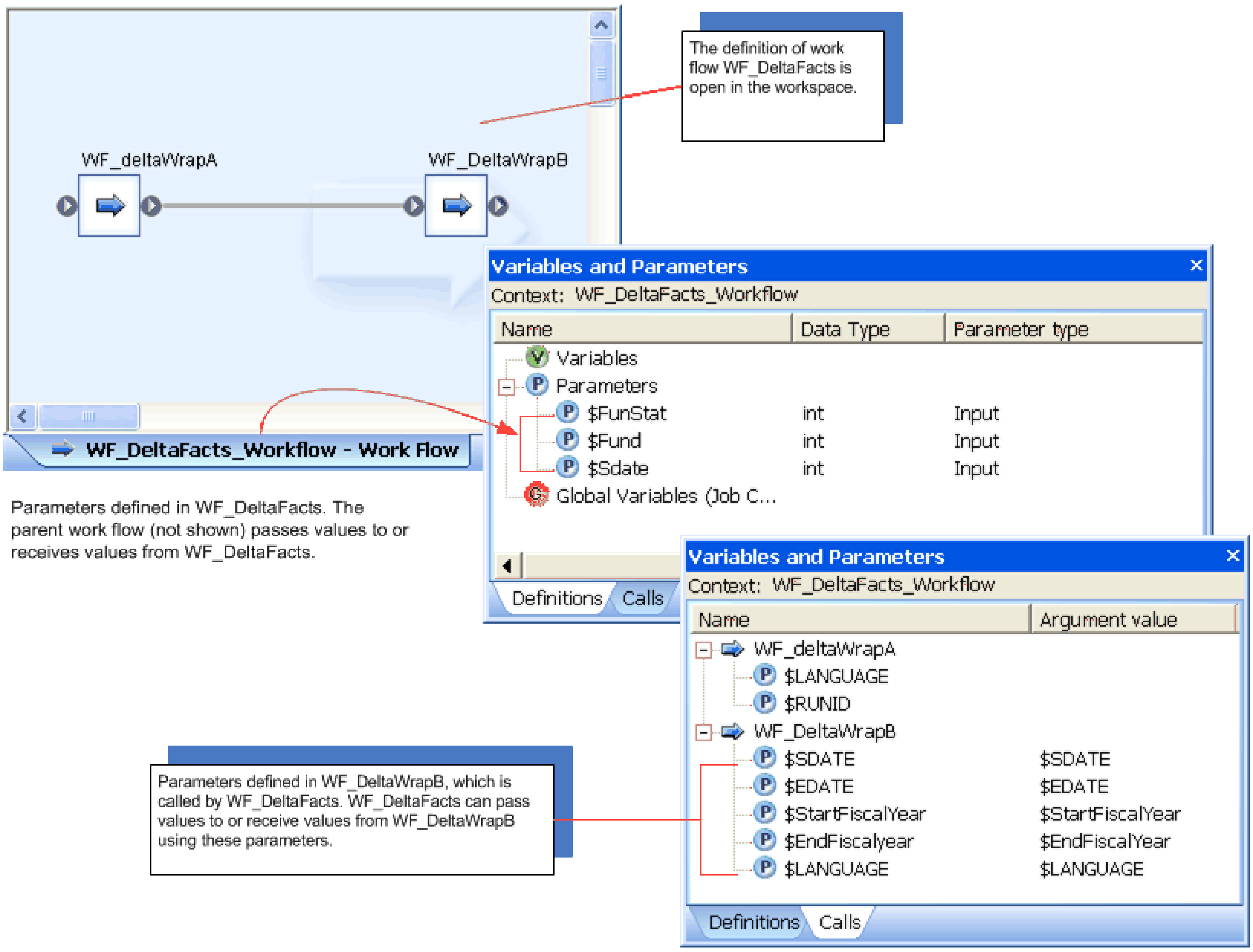
For the input parameter type, values in the *Calls* tab can be constants, variables, or another parameter.

For the output or input/output parameter type, values in the *Calls* tab can be variables or parameters.

Values in the *Calls* tab must also use:

- The same data type as the variable if they are placed inside an input or input/output parameter type, and a compatible data type if they are placed inside an output parameter type.
- Scripting language rules and syntax.

The following illustration shows the relationship between an open work flow called DeltaFacts, the *Context* box in the Variables and Parameters window, and the content in the *Definition* and *Calls* tabs.



The definition of work flow WF_DeltaFacts is open in the workspace.

Parameters defined in WF_DeltaFacts. The parent work flow (not shown) passes values to or receives values from WF_DeltaFacts.

Variables and Parameters

Context: WF_DeltaFacts_Workflow

Name	Data Type	Parameter type
Variables		
Parameters		
$FunStat	int	Input
$Fund	int	Input
$Sdate	int	Input
Global Variables (Job C...		

Variables and Parameters

Context: WF_DeltaFacts_Workflow

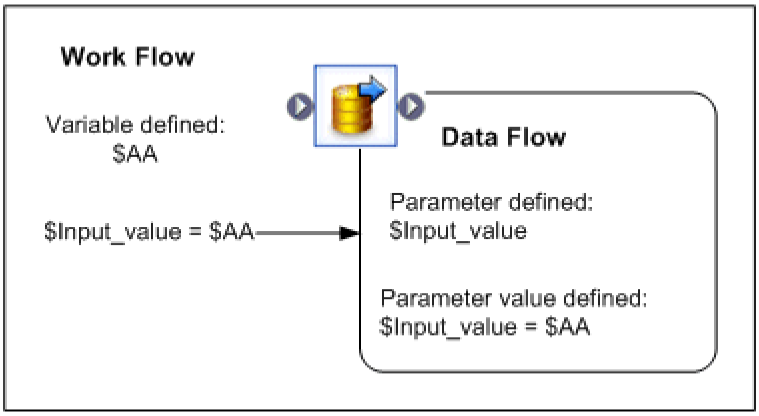
Name	Argument value
WF_deltaWrapA	
$LANGUAGE	
$RUNID	
WF_DeltaWrapB	
$SDATE	$SDATE
$EDATE	$EDATE
$StartFiscalYear	$StartFiscalYear
$EndFiscalyear	$EndFiscalYear
$LANGUAGE	$LANGUAGE

Parameters defined in WF_DeltaWrapB, which is called by WF_DeltaFacts. WF_DeltaFacts can pass values to or receive values from WF_DeltaWrapB using these parameters.

2.13.3 Using local variables and parameters

To pass a local variable to another object, define the local variable, then from the calling object, create a parameter and map the parameter to the local variable by entering a parameter value.

For example, to use a local variable inside a data flow, define the variable in a parent work flow and then pass the value of the variable as a parameter of the data flow.


Work Flow
Variable defined: $AA
$Input_value = $AA
Data Flow
Parameter defined: $Input_value
Parameter value defined: $Input_value = $AA

2.13.3.1 Parameters

Parameters can be defined to:

- Pass their values into and out of work flows
- Pass their values into data flows

Each parameter is assigned a type: input, output, or input/output. The value passed by the parameter can be used by any object called by the work flow or data flow.

> Note
>
> You can also create local variables and parameters for use in custom functions.

Related Information

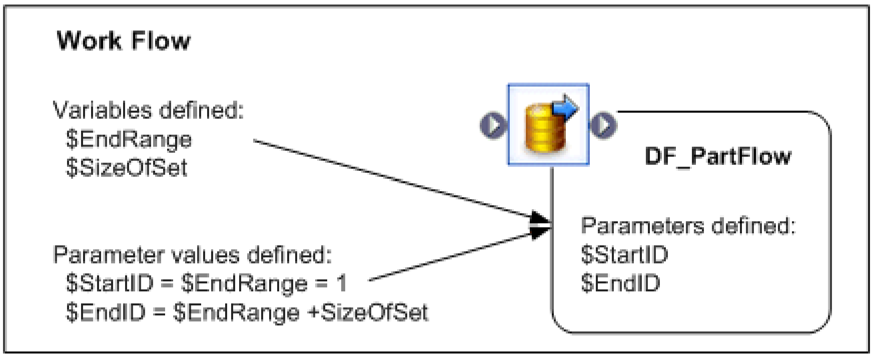
Reference Guide: Custom functions [page 1694]

2.13.3.2 Passing values into data flows

You can use a value passed as a parameter into a data flow to control the data transformed in the data flow. For example, the data flow DF_PartFlow processes daily inventory values. It can process all of the part numbers in use or a range of part numbers based on external requirements such as the range of numbers processed most recently.

If the work flow that calls DF_PartFlow records the range of numbers processed, it can pass the end value of the range `$EndRange` as a parameter to the data flow to indicate the start value of the range to process next.

The software can calculate a new end value based on a stored number of parts to process each time, such as `$SizeOfSet`, and pass that value to the data flow as the end value. A query transform in the data flow uses the parameters passed in to filter the part numbers extracted from the source.



The data flow could be used by multiple calls contained in one or more work flows to perform the same task on different part number ranges by specifying different parameters for the particular calls.

2.13.3.3 To define a local variable

1. Click the name of the job or work flow in the project area or workspace, or double-click one from the object library.
2. Click *Tools* > *Variables* .

   The *Variables and Parameters* window appears.
3. From the *Definitions* tab, select *Variables*.
4. Right-click and select *Insert*. Or, to insert a variable in a specific position in the list, select an existing variable, right-click, and select *Insert Before* or *Insert After*.

   A new variable appears (for example, `$NewVariable0`). A focus box appears around the name cell and the cursor shape changes to an arrow with a yellow pencil.
5. To edit the name of the new variable, click the name cell.

   The name can include alphanumeric characters or underscores (_), but cannot contain blank spaces. Always begin the name with a dollar sign ($).
6. Click the data type cell for the new variable and select the appropriate data type from the drop-down list.
7. To specify the order of the variable in the list, select the variable, right-click, and select *Move Up* or *Move Down*.
8. Close the *Variables and Parameters* window.

2.13.3.4 To replicate a local variable

1. Click the name of the job or work flow in the project area or workspace, or double-click one from the object library.
2. Click *Tools* > *Variables* .

   The *Variables and Parameters* window appears.
3. From the *Definitions* tab, select *Variables*.
4. Select an existing variable, right-click, and select *Replicate*.

A properties window opens and displays a new variable with the same properties as the replicated variable (for example, `$Copy_1_NewVariable1`).

5. Edit the name of the new variable.

   The name can include alphanumeric characters or underscores (_), but cannot contain blank spaces. Always begin the name with a dollar sign ($).
6. Select the appropriate data type from the drop-down list.
7. Click *OK*.
8. To specify the order of the variable in the list, select the variable, right-click, and select *Move Up* or *Move Down*.
9. Close the *Variables and Parameters* window.

2.13.3.5 Defining parameters

There are two steps for setting up a parameter for a work flow or data flow:

- Add the parameter definition to the flow.
- Set the value of the parameter in the flow call.

2.13.3.5.1 To add a parameter to a work flow or data flow

1. Click the name of the work flow or data flow.
2. Click ▶ *Tools* › *Variables* ◀.

   The *Variables and Parameters* window appears.
3. Go to the *Definition* tab.
4. Select *Parameters*.
5. Right-click and select *Insert*. Or, to insert a parameter in a specific position in the list, select an existing parameter, right-click, and select *Insert Before* or *Insert After*.

   A new parameter appears (for example, `$NewParameter0`). A focus box appears and the cursor shape changes to an arrow with a yellow pencil.
6. To edit the name of the new variable, click the name cell.

   The name can include alphanumeric characters or underscores (_), but cannot contain blank spaces. Always begin the name with a dollar sign ($).
7. Click the data type cell for the new parameter and select the appropriate data type from the drop-down list.

   If the parameter is an input or input/output parameter, it must have the same data type as the variable; if the parameter is an output parameter type, it must have a compatible data type.
8. Click the parameter type cell and select the parameter type (input, output, or input/output).
9. To specify the order of the parameter in the list, select the parameter, right-click, and select *Move Up* or *Move Down*.
10. Close the *Variables and Parameters* window.

2.13.3.5.2 To set the value of the parameter in the flow call

1. Open the calling job, work flow, or data flow.
2. Click ▸ *Tools* ▸ *Variables* ◢ to open the *Variables and Parameters* window.
3. Select the *Calls* tab.

   The Calls tab shows all the objects that are called from the open job, work flow, or data flow.
4. Click the *Argument Value* cell.
   A focus box appears and the cursor shape changes to an arrow with a yellow pencil.
5. Enter the expression the parameter will pass in the cell.

   If the parameter type is input, then its value can be an expression that contains a constant (for example, 0, 3, or 'string1'), a variable, or another parameter (for example, `$startID` or `$parm1`).

   If the parameter type is output or input/output, then the value must be a variable or parameter. The value cannot be a constant because, by definition, the value of an output or input/output parameter can be modified by any object within the flow.

   To indicate special values, use the following syntax:

Value type	Special syntax
Variable	**`<$variable_name>`**
String	**`<'string>`**'

2.13.4 Using global variables

Global variables are global within a job. Setting parameters is not necessary when you use global variables. However, once you use a name for a global variable in a job, that name becomes reserved for the job. Global variables are exclusive within the context of the job in which they are created.

2.13.4.1 Creating global variables

Define variables in the Variables and Parameter window.

2.13.4.1.1 To create a global variable

1. Click the name of a job in the project area or double-click a job from the object library.
2. Click ▸ *Tools* ▸ *Variables* ◢.
   The *Variables and Parameters* window appears.

3. From the *Definitions* tab, select *Global Variables*.
4. Right-click *Global Variables* and select *Insert*. Or, to insert a variable in a specific position in the list, select an existing variable, right-click, and select *Insert Before* or *Insert After*.

   A new global variable appears (for example, `$NewJobGlobalVariable0`). A focus box appears and the cursor shape changes to an arrow with a yellow pencil.
5. To edit the name of the new variable, click the name cell.

   The name can include alphanumeric characters or underscores (_), but cannot contain blank spaces. Always begin the name with a dollar sign ($).
6. Click the data type cell for the new variable and select the appropriate data type from the drop-down list.
7. To specify the order of the variable in the list, select the variable, right-click, and select *Move Up* or *Move Down*.
8. Close the *Variables and Parameters* window.

2.13.4.1.2 To replicate a global variable

1. Click the name of a job in the project area or double-click a job from the object library.
2. Click ▸ *Tools* ▸ *Variables* ◂.

   The *Variables and Parameters* window appears.
3. From the *Definitions* tab, select *Global Variables*.
4. Select an existing variable, right-click, and select *Replicate*.

   A properties window opens and displays a new variable with the same properties as the replicated variable (for example, `$Copy_1_NewJobGlobalVariable1`).
5. Edit the name of the new global variable.

   The name can include alphanumeric characters or underscores (_), but cannot contain blank spaces. Always begin the name with a dollar sign ($).
6. Select the appropriate data type from the drop-down list.
7. Click *OK*.
8. To specify the order of the variable in the list, select the variable, right-click, and select *Move Up* or *Move Down*.
9. Close the *Variables and Parameters* window.

2.13.4.2 Viewing global variables

Global variables, defined in a job, are visible to those objects relative to that job. A global variable defined in one job is not available for modification or viewing from another job.

You can view global variables from the Variables and Parameters window (with an open job in the work space) or from the Properties dialog of a selected job.

2.13.4.2.1 To view global variables in a job from the Properties dialog

1. In the object library, select the *Jobs* tab.
2. Right-click the job whose global variables you want to view and select *Properties*.
3. Click the *Global Variable* tab.

   Global variables appear on this tab.

2.13.4.3 Setting global variable values

In addition to setting a variable inside a job using an initialization script, you can set and maintain global variable values outside a job. Values set outside a job are processed the same way as those set in an initialization script. However, if you set a value for the same variable both inside and outside a job, the internal value will override the external job value.

Values for global variables can be set outside a job:

- As a job property
- As an execution or schedule property

Global variables without defined values are also allowed. They are read as NULL.

All values defined as job properties are shown in the Properties and the Execution Properties dialogs of the Designer and in the Execution Options and Schedule pages of the Administrator. By setting values outside a job, you can rely on these dialogs for viewing values set for global variables and easily edit values when testing or scheduling a job.

> Note
>
> You cannot pass global variables as command line arguments for real-time jobs.

2.13.4.3.1 To set a global variable value as a job property

1. Right-click a job in the object library or project area.
2. Click *Properties*.
3. Click the *Global Variable* tab.

   All global variables created in the job appear.
4. To filter the displayed global variables by name, enter part of the name in the *Filter* box.
5. Enter values for the global variables in this job.

   You can use any statement used in a script with this option.
6. To set multiple global variable values, use the Control button to select the variables, right-click, and select *Update Value*. Enter the value in the Smart Editor and click *OK*.

7. Click *OK*.

   The software saves values in the repository as job properties.

   You can also view and edit these default values in the Execution Properties dialog of the Designer and in the Execution Options and Schedule pages of the Administrator. This allows you to override job property values at run-time.

Related Information

Reference Guide: Scripting Language [page 1709]

2.13.4.3.2 To set a global variable value as an execution property

1. Execute a job from the Designer, or execute or schedule a batch job from the Administrator.

   > Note
   >
   > For testing purposes, you can execute real-time jobs from the Designer in test mode. Make sure to set the execution properties for a real-time job.

2. View the global variables in the job and their default values (if available).
3. Edit values for global variables as desired.
4. If you are using the Designer, click *OK*. If you are using the Administrator, click *Execute* or *Schedule*.

   The job runs using the values you enter. Values entered as execution properties are not saved. Values entered as schedule properties are saved but can only be accessed from within the Administrator.

2.13.4.3.3 Automatic ranking of global variable values in a job

Using the methods described in the previous section, if you enter different values for a single global variable, the software selects the highest ranking value for use in the job. A value entered as a job property has the lowest rank. A value defined inside a job has the highest rank.

- If you set a global variable value as both a job and an execution property, the execution property value overrides the job property value and becomes the default value for the current job run. You cannot save execution property global variable values.
  For example, assume that a job, JOB_Test1, has three global variables declared: $YEAR, $MONTH, and $DAY. Variable $YEAR is set as a job property with a value of 2003.
  For the job run, you set variables $MONTH and $DAY as execution properties to values 'JANUARY' and 31 respectively. The software executes a list of statements which includes default values for JOB_Test1:

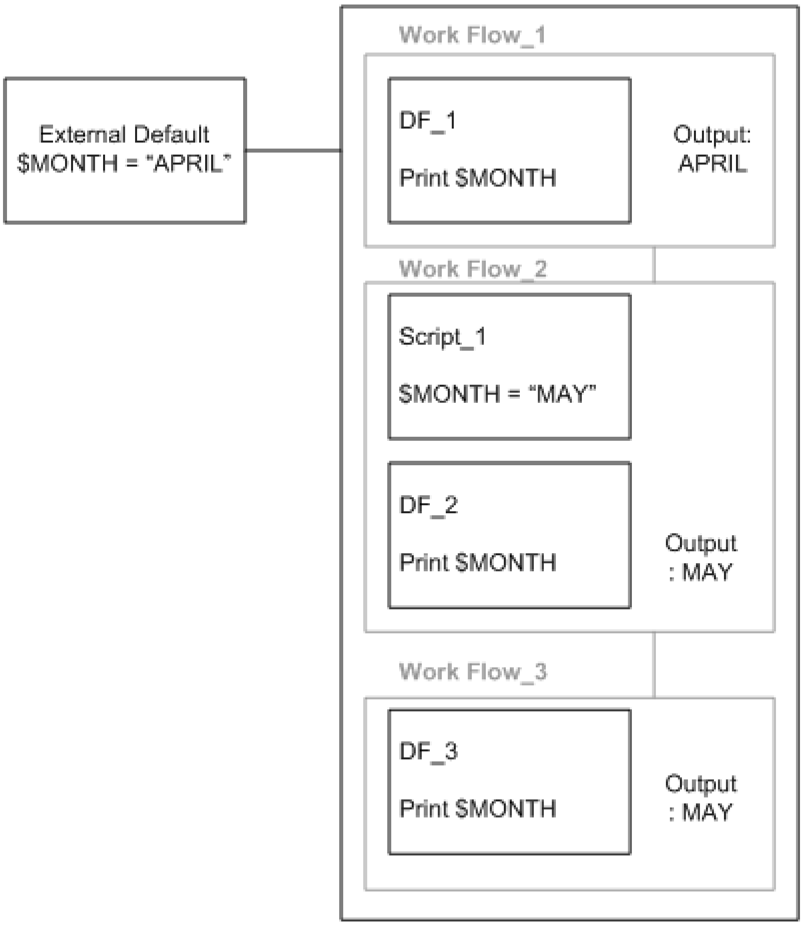
  ```
  $YEAR=2003;
  $MONTH='JANUARY';
  $DAY=31;
  ```

For the second job run, if you set variables $YEAR and $MONTH as execution properties to values 2002 and 'JANUARY' respectively, then the statement $YEAR=2002 will replace $YEAR=2003. The software executes the following list of statements:

```
$YEAR=2002;
$MONTH='JANUARY';
```

> **Note**
>
> In this scenario, $DAY is not defined and the software reads it as NULL. You set $DAY to 31 during the first job run; however, execution properties for global variable values are not saved.

- If you set a global variable value for both a job property and a schedule property, the schedule property value overrides the job property value and becomes the external, default value for the current job run.
  The software saves schedule property values in the repository. However, these values are only associated with a job schedule, not the job itself. Consequently, these values are viewed and edited from within the Administrator.
- A global variable value defined inside a job always overrides any external values. However, the override does not occur until the software attempts to apply the external values to the job being processed with the internal value. Up until that point, the software processes execution, schedule, or job property values as default values.
  For example, suppose you have a job called JOB_Test2 that has three work flows, each containing a data flow. The second data flow is inside a work flow that is preceded by a script in which $MONTH is defined as 'MAY'. The first and third data flows have the same global variable with no value defined. The execution property $MONTH = 'APRIL' is the global variable value.
  In this scenario, 'APRIL' becomes the default value for the job. 'APRIL' remains the value for the global variable until it encounters the other value for the same variable in the second work flow. Since the value in the script is inside the job, 'MAY' overrides 'APRIL' for the variable $MONTH. The software continues the processing the job with this new value.



2.13.4.3.4 Advantages to setting values outside a job

While you can set values inside jobs, there are advantages to defining values for global variables outside a job.

For example, values defined as job properties are shown in the Properties and the Execution Properties dialogs of the Designer and in the Execution Options and Schedule pages of the Administrator. By setting values outside a job, you can rely on these dialogs for viewing all global variables and their values. You can also easily edit them for testing and scheduling.

In the Administrator, you can set global variable values when creating or editing a schedule without opening the Designer. For example, use global variables as file names and start and end dates.

2.13.5 Local and global variable rules

When defining local or global variables, consider rules for:

- Naming
- Replicating jobs and work flows

- Importing and exporting

2.13.5.1 Naming

- Local and global variables must have unique names within their job context.
- Any name modification to a global variable can only be performed at the job level.

2.13.5.2 Replicating jobs and work flows

- When you replicate all objects, the local and global variables defined in that job context are also replicated.
- When you replicate a data flow or work flow, all parameters and local and global variables are also replicated. However, you must validate these local and global variables within the job context in which they were created. If you attempt to validate a data flow or work flow containing global variables without a job, Data Services reports an error.

2.13.5.3 Importing and exporting

- When you export a job object, you also export all local and global variables defined for that job.
- When you export a lower-level object (such as a data flow) without the parent job, the global variable is not exported. Only the call to that global variable is exported. If you use this object in another job without defining the global variable in the new job, a validation error will occur.

2.13.6 Environment variables

You can use system-environment variables inside jobs, work flows, or data flows. The get_env, set_env, and is_set_env functions provide access to underlying operating system variables that behave as the operating system allows.

You can temporarily set the value of an environment variable inside a job, work flow or data flow. Once set, the value is visible to all objects in that job.

Use the get_env, set_env, and is_set_env functions to set, retrieve, and test the values of environment variables.

Related Information

Reference Guide: Functions and Procedures [page 1511]

2.13.7 Setting file names at run-time using variables

You can set file names at runtime by specifying a variable as the file name.

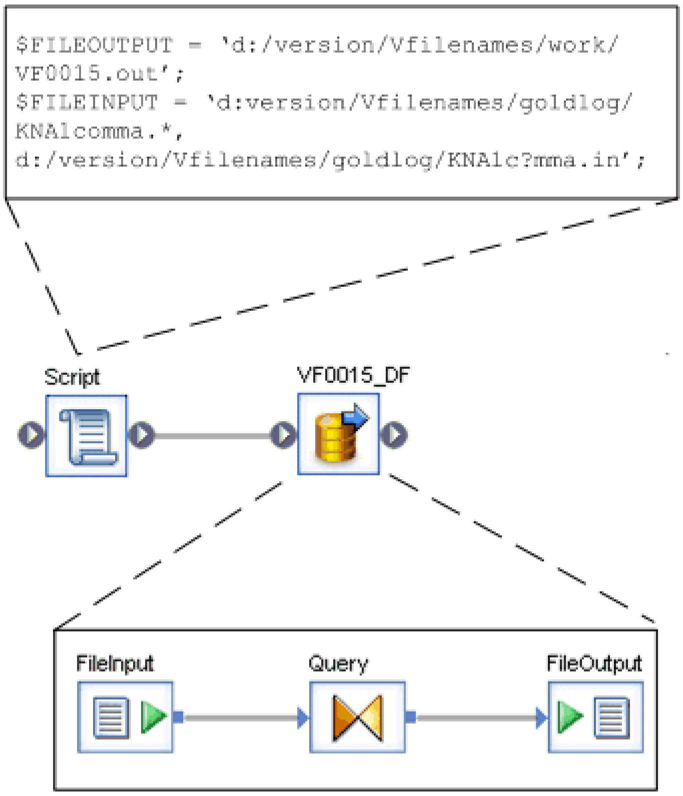
Variables can be used as file names for:

- The following sources and targets:
  - Flat files
  - XML files and messages
  - IDoc files and messages (in an SAP environment)
- The `lookup_ext` function (for a flat file used as a lookup table parameter)

2.13.7.1 To use a variable in a flat file name

1. Create a local or global variable using the Variables and Parameters window.
2. Create a script to set the value of a local or global variable, or call a system environment variable.
3. Declare the variable in the file format editor or in the Function editor as a `lookup_ext` parameter.
   - When you set a variable value for a flat file, specify both the file name and the directory name. Enter the variable in the *File(s)* property under *Data File(s)* in the File Format Editor. You cannot enter a variable in the *Root directory* property.
   - For lookups, substitute the path and file name in the *Lookup table* box in the `lookup_ext` function editor with the variable name.

The following figure shows how you can set values for variables in flat file sources and targets in a script.



```
$FILEOUTPUT = 'd:/version/Vfilenames/work/
VF0015.out';
$FILEINPUT = 'd:version/Vfilenames/goldlog/
KNA1comma.*,
d:/version/Vfilenames/goldlog/KNA1c?mma.in';
```

When you use variables as sources and targets, you can also use multiple file names and wild cards. Neither is supported when using variables in the `lookup_ext` function.

The figure above provides an example of how to use multiple variable names and wild cards. Notice that the `$FILEINPUT` variable includes two file names (separated by a comma). The two names (`KNA1comma.*` and `KNA1c?mma.in`) also make use of the wild cards (`*` and `?`) supported by the software.

Related Information

Reference Guide: lookup_ext [page 1605]
Reference Guide: Data Services Scripting Language [page 1709]

2.13.8 Substitution parameters

2.13.8.1 Overview of substitution parameters

Substitution parameters are useful when you want to export and run a job containing constant values in a specific environment. For example, if you create a job that references a unique directory on your local computer and you export that job to another computer, the job will look for the unique directory in the new environment. If that directory doesn't exist, the job won't run.

Instead, by using a substitution parameter, you can easily assign a value for the original, constant value in order to run the job in the new environment. After creating a substitution parameter value for the directory in your environment, you can run the job in a different environment and all the objects that reference the original directory will automatically use the value. This means that you only need to change the constant value (the original directory name) in one place (the substitution parameter) and its value will automatically propagate to all objects in the job when it runs in the new environment.

You can configure a group of substitution parameters for a particular run-time environment by associating their constant values under a substitution parameter configuration.

2.13.8.1.1 Substitution parameters versus global variables

Substitution parameters differ from global variables in that they apply at the repository level. Global variables apply only to the job in which they are defined. You would use a global variable when you do not know the value prior to execution and it needs to be calculated in the job. You would use a substitution parameter for constants that do not change during execution. A substitution parameter defined in a given local repository is available to all the jobs in that repository. Therefore, using a substitution parameter means you do not need to define a global variable in each job to parameterize a constant value.

The following table describes the main differences between global variables and substitution parameters.

Global variables	Substitution parameters
Defined at the job level	Defined at the repository level
Cannot be shared across jobs	Available to all jobs in a repository
Data-type specific	No data type (all strings)
Value can change during job execution	Fixed value set prior to execution of job (constants)

However, you can use substitution parameters in all places where global variables are supported, for example:

- Query transform WHERE clauses
- Mappings
- SQL transform SQL statement identifiers
- Flat-file options
- User-defined transforms
- Address cleanse transform options
- Matching thresholds

2.13.8.1.2 Using substitution parameters

You can use substitution parameters in expressions, SQL statements, option fields, and constant strings. For example, many options and expression editors include a drop-down menu that displays a list of all the available substitution parameters.

The software installs some default substitution parameters that are used by some Data Quality transforms. For example, the USA Regulatory Address Cleanse transform uses the following built-in substitution parameters:

- $$RefFilesAddressCleanse defines the location of the address cleanse directories.
- $$ReportsAddressCleanse (set to Yes or No) enables data collection for creating reports with address cleanse statistics. This substitution parameter provides one location where you can enable or disable that option for all jobs in the repository.

Other examples of where you can use substitution parameters include:

- In a script, for example:

  ```
  Print('Data read in : [$$FilePath]'); or Print('[$$FilePath]');
  ```
- In a file format, for example with `[$$FilePath]/file.txt` as the file name

2.13.8.2 Using the Substitution Parameter Editor

Open the *Substitution Parameter Editor* from the Designer by selecting *Tools* › *Substitution Parameter Configurations*. Use the Substitution Parameter editor to do the following tasks:

- Add and define a substitution parameter by adding a new row in the editor.
- For each substitution parameter, use right-click menus and keyboard shortcuts to Cut, Copy, Paste, Delete, and Insert parameters.
- Change the order of substitution parameters by dragging rows or using the Cut, Copy, Paste, and Insert commands.
- Add a substitution parameter configuration by clicking the *Create New Substitution Parameter Configuration* icon in the toolbar.
- Duplicate an existing substitution parameter configuration by clicking the *Create Duplicate Substitution Parameter Configuration* icon.
- Rename a substitution parameter configuration by clicking the *Rename Substitution Parameter Configuration* icon.
- Delete a substitution parameter configuration by clicking the *Delete Substitution Parameter Configuration* icon.
- Reorder the display of configurations by clicking the *Sort Configuration Names in Ascending Order* and *Sort Configuration Names in Descending Order* icons.
- Move the default configuration so it displays next to the list of substitution parameters by clicking the *Move Default Configuration To Front* icon.
- Change the default configuration.

Related Information

Adding and defining substitution parameters [page 417]

2.13.8.2.1 Naming substitution parameters

When you name and define substitution parameters, use the following rules:

- The name prefix is two dollar signs $$ (global variables are prefixed with one dollar sign). When adding new substitution parameters in the Substitution Parameter Editor, the editor automatically adds the prefix.
- When typing names in the Substitution Parameter Editor, do not use punctuation (including quotes or brackets) except underscores. The following characters are not allowed:

  ```
  ,: / ' \ " = < > + | - * % ; \t [ ] ( ) \r \n $ ] +
  ```

- You can type names directly into fields, column mappings, transform options, and so on. However, you must enclose them in square brackets, for example [$$SamplesInstall].
- Names can include any alpha or numeric character or underscores but cannot contain spaces.
- Names are not case sensitive.
- The maximum length for most repository types is 256 (MySQL is 64 and MS SQL server is 128).
- Names must be unique within the repository.

2.13.8.2.2 Adding and defining substitution parameters

1. In the Designer, open the Substitution Parameter Editor by selecting ▶ *Tools* ▶ *Substitution Parameter Configurations* ◀.
2. The first column lists the substitution parameters available in the repository. To create a new one, double-click in a blank cell (a pencil icon will appear in the left) and type a name. The software automatically adds a double dollar-sign prefix ($$) to the name when you navigate away from the cell.
3. The second column identifies the name of the first configuration, by default *Configuration1* (you can change configuration names by double-clicking in the cell and retyping the name). Double-click in the blank cell next to the substitution parameter name and type the constant value that the parameter represents in that configuration. The software applies that value when you run the job.
4. To add another configuration to define a second value for the substitution parameter, click the *Create New Substitution Parameter Configuration* icon on the toolbar.
5. Type a unique name for the new substitution parameter configuration.
6. Enter the value the substitution parameter will use for that configuration.

You can now select from one of the two substitution parameter configurations you just created.

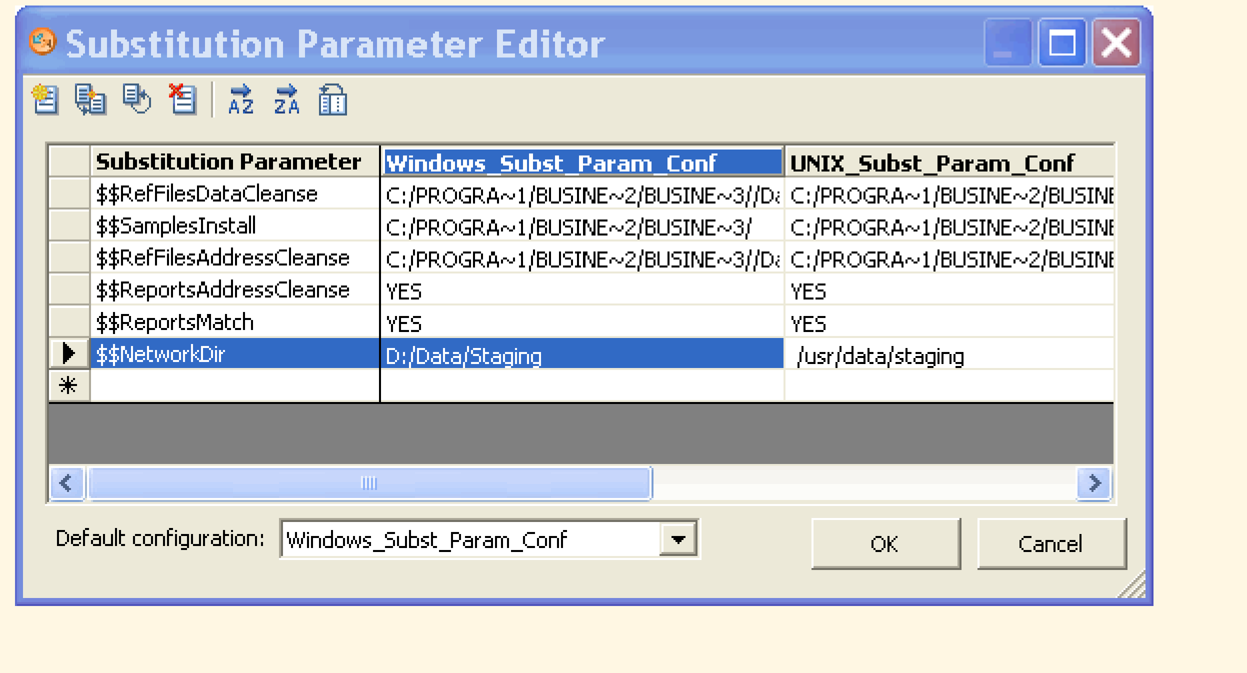
To change the default configuration that will apply when you run jobs, select it from the drop-down list box at the bottom of the window.

You can also export these substitution parameter configurations for use in other environments.

Example

In the following example, the substitution parameter $$NetworkDir has the value D:/Data/Staging in the configuration named Windows_Subst_Param_Conf and the value /usr/data/staging in the UNIX_Subst_Param_Conf configuration.

Notice that each configuration can contain multiple substitution parameters.



Substitution Parameter	Windows_Subst_Param_Conf	UNIX_Subst_Param_Conf
$$RefFilesDataCleanse	C:/PROGRA~1/BUSINE~2/BUSINE~3//Da	C:/PROGRA~1/BUSINE~2/BUSINE
$$SamplesInstall	C:/PROGRA~1/BUSINE~2/BUSINE~3/	C:/PROGRA~1/BUSINE~2/BUSINE
$$RefFilesAddressCleanse	C:/PROGRA~1/BUSINE~2/BUSINE~3//Da	C:/PROGRA~1/BUSINE~2/BUSINE
$$ReportsAddressCleanse	YES	YES
$$ReportsMatch	YES	YES
$$NetworkDir	D:/Data/Staging	/usr/data/staging

Default configuration: Windows_Subst_Param_Conf

Related Information

Naming substitution parameters [page 416]
Exporting and importing substitution parameters [page 421]

2.13.8.3 Associating a substitution parameter configuration with a system configuration

A system configuration groups together a set of datastore configurations and a substitution parameter configuration. A substitution parameter configuration can be associated with one or more system configurations. For example, you might create one system configuration for your local system and a different system configuration for another system. Depending on your environment, both system configurations might point to the same substitution parameter configuration or each system configuration might require a different substitution parameter configuration.

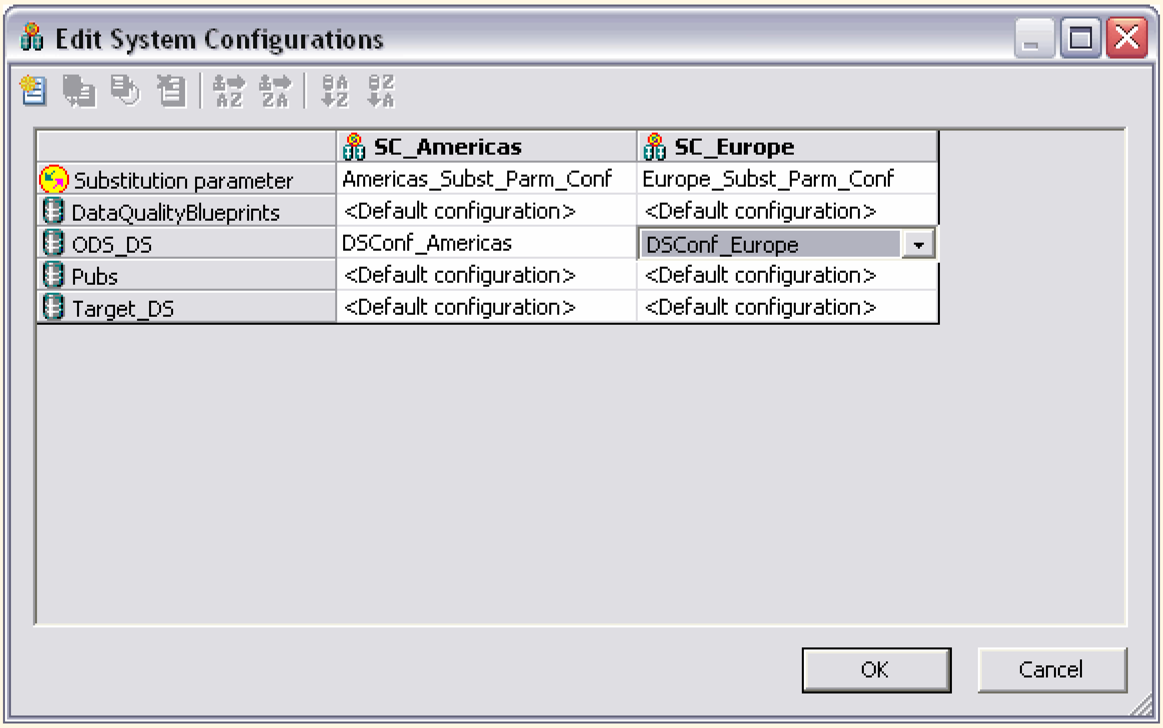
At job execution time, you can set the system configuration and the job will execute with the values for the associated substitution parameter configuration.

To associate a substitution parameter configuration with a new or existing system configuration:

1. In the Designer, open the System Configuration Editor by selecting *Tools* > *System Configurations*.
2. Optionally create a new system configuration.
3. Under the desired system configuration name, select a substitution parameter configuration to associate with the system configuration.
4. Click *OK*.

Example

The following example shows two system configurations, Americas and Europe. In this case, there are substitution parameter configurations for each region (Europe_Subst_Parm_Conf and Americas_Subst_Parm_Conf). Each substitution parameter configuration defines where the data source files are located for that region, for example D:/Data/Americas and D:/Data/Europe. Select the appropriate substitution parameter configuration and datastore configurations for each system configuration.



	SC_Americas	SC_Europe
Substitution parameter	Americas_Subst_Parm_Conf	Europe_Subst_Parm_Conf
DataQualityBlueprints	<Default configuration>	<Default configuration>
ODS_DS	DSConf_Americas	DSConf_Europe
Pubs	<Default configuration>	<Default configuration>
Target_DS	<Default configuration>	<Default configuration>

Related Information

Defining a system configuration [page 252]

2.13.8.4 Overriding a substitution parameter in the Administrator

In the Administrator, you can override the substitution parameters, or select a system configuration to specify a substitution parameter configuration, on four pages:

- Execute Batch Job
- Schedule Batch Job

- Export Execution Command
- Real-Time Service Configuration

For example, the Execute Batch Job page displays the name of the selected system configuration, the substitution parameter configuration, and the name of each substitution parameter and its value.

To override a substitution parameter:

1. Select the appropriate system configuration.
2. Under *Substitution Parameters*, click *Add Overridden Parameter*, which displays the available substitution parameters.
3. From the drop-down list, select the substitution parameter to override.
4. In the second column, type the override value. Enter the value as a string without quotes (in contrast with Global Variables).
5. Execute the job.

2.13.8.5 Executing a job with substitution parameters

To see the details of how substitution parameters are being used in the job during execution in the Designer trace log:

1. Right-click the job name and click *Properties*.
2. Click the *Trace* tab.
3. For the *Trace Assemblers* option, set the value to *Yes.*
4. Click *OK.*

When you execute a job from the Designer, the Execution Properties window displays. You have the following options:

- On the *Execution Options* tab from the *System configuration* drop-down menu, optionally select the system configuration with which you want to run the job. If you do not select a system configuration, the software applies the default substitution parameter configuration as defined in the Substitution Parameter Editor.
  You can click *Browse* to view the *Select System Configuration* window in order to see the substitution parameter configuration associated with each system configuration. The *Select System Configuration* is read-only. If you want to change a system configuration, click ▸ *Tools* ▸ *System Configurations* ◂.
- You can override the value of specific substitution parameters at run time. Click the *Substitution Parameter* tab, select a substitution parameter from the Name column, and enter a value by double-clicking in the Value cell.
  To override substitution parameter values when you start a job via a Web service, see the *Integrator Guide*.

Related Information

Associating a substitution parameter configuration with a system configuration [page 418]
Overriding a substitution parameter in the Administrator [page 419]

2.13.8.6 Exporting and importing substitution parameters

Substitution parameters are stored in a local repository along with their configured values. The software does not include substitution parameters as part of a regular export.You can, however, export substitution parameters and configurations to other repositories by exporting them to a file and then importing the file to another repository.

2.13.8.6.1 Exporting substitution parameters

1. Right-click in the local object library and select *Repository* ▸ *Export Substitution Parameter Configurations*.
2. Select the check box in the *Export* column for the substitution parameter configurations to export.
3. *Save* the file.
   The software saves it as a text file with an .atl extension.

2.13.8.6.2 Importing substitution parameters

The substitution parameters must have first been exported to an ATL file.

Be aware of the following behaviors when importing substitution parameters:

- The software adds any new substitution parameters and configurations to the destination local repository.
- If the repository has a substitution parameter with the same name as in the exported file, importing will overwrite the parameter's value. Similarly, if the repository has a substitution parameter configuration with the same name as the exported configuration, importing will overwrite all the parameter values for that configuration.

1. In the Designer, right-click in the object library and select *Repository* ▸ *Import from file*.
2. Browse to the file to import.
3. Click *OK*.

Related Information

Exporting substitution parameters [page 421]

2.14 Executing Jobs

This section contains an overview of the software job execution, steps to execute jobs, debug errors, and change job server options.

2.14.1 Overview of job execution

You can run jobs in three different ways. Depending on your needs, you can configure:

- *Immediate jobs*
  The software initiates both batch and real-time jobs and runs them immediately from within the Designer. For these jobs, both the Designer and designated Job Server (where the job executes, usually many times on the same machine) must be running. You will most likely run immediate jobs only during the development cycle.
- *Scheduled jobs*
  Batch jobs are scheduled. To schedule a job, use the Administrator or use a third-party scheduler.
  When jobs are scheduled by third-party software:
  - The job initiates outside of the software.
  - The job operates on a batch job (or shell script for UNIX) that has been exported from the software.

  When a job is invoked by a third-party scheduler:
  - The corresponding Job Server must be running.
  - The Designer does not need to be running.
- *Services*
  Real-time jobs are set up as services that continuously listen for requests from an Access Server and process requests on-demand as they are received. Use the Administrator to create a service from a real-time job.

2.14.2 Preparing for job execution

2.14.2.1 Validating jobs and job components

You can also explicitly validate jobs and their components as you create them by:

	Clicking the *Validate All* button from the toolbar (or choosing *Validate* > *All Objects in View* from the *Debug* menu). This command checks the syntax of the object definition for the active workspace and for all objects that are called from the active workspace view recursively.
	Clicking the *Validate Current View* button from the toolbar (or choosing *Validate* > *Current View* from the *Debug* menu). This command checks the syntax of the object definition for the active workspace.

You can set the Designer options (*Tools* > *Options* > *Designer* > *General*) to validate jobs started in Designer before job execution. The default is not to validate.

The software also validates jobs before exporting them.

If during validation the software discovers an error in an object definition, it opens a dialog box indicating that an error exists, then opens the Output window to display the error.

If there are errors, double-click the error in the Output window to open the editor of the object containing the error.

If you are unable to read the complete error text in the window, you can access additional information by right-clicking the error listing and selecting *View* from the context menu.

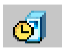
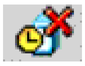
Error messages have these levels of severity:

Severity	Description
Information ⓘ	Informative message only—does not prevent the job from running. No action is required.
Warning ⚠	The error is not severe enough to stop job execution, but you might get unexpected results. For example, if the data type of a source column in a transform within a data flow does not match the data type of the target column in the transform, the software alerts you with a warning message.
Error ✖	The error is severe enough to stop job execution. You must fix the error before the job will execute.

2.14.2.2 Ensuring that the Job Server is running

Before you execute a job (either as an immediate or scheduled task), ensure that the Job Server is associated with the repository where the client is running.

When the Designer starts, it displays the status of the Job Server for the repository to which you are connected.

Icon	Description
	Job Server is running
	Job Server is inactive

The name of the active Job Server and port number appears in the status bar when the cursor is over the icon.

2.14.2.3 Setting job execution options

Options for jobs include Debug and Trace. Although these are object options—they affect the function of the object—they are located in either the Property or the Execution window associated with the job.

Execution options for jobs can either be set for a single instance or as a default value.

- The right-click *Execute* menu sets the options for a single execution only and overrides the default settings.
- The right-click *Properties* menu sets the default settings.

2.14.2.3.1 To set execution options for every execution of the job

1. From the *Project* area, right-click the job name and choose *Properties*.
2. Select options on the Properties window.

Related Information

Viewing and changing object properties [page 190]
Reference Guide: Parameters [page 840]
Reference Guide: Trace properties [page 843]
Setting global variable values [page 408]

2.14.3 Executing jobs as immediate tasks

Immediate or "on demand" tasks are initiated from the Designer. Both the Designer and Job Server must be running for the job to execute.

2.14.3.1 To execute a job as an immediate task

1. In the project area, select the job name.
2. Right-click and choose *Execute*.

   The software prompts you to save any objects that have changes that have not been saved.
3. The next step depends on whether you selected the *Perform complete validation before job execution* check box in the Designer Options:
   - If you have not selected this check box, a window opens showing execution properties (debug and trace) for the job. Proceed to the next step.
   - If you have selected this check box, the software validates the job before it runs. You must correct any serious errors before the job will run. There might also be warning messages—for example, messages indicating that date values will be converted to datetime values. Correct them if you want (they will not prevent job execution) or click *OK* to continue. After the job validates, a window opens showing the execution properties (debug and trace) for the job.
4. Set the execution properties.

   You can choose the Job Server that you want to process this job, datastore profiles for sources and targets if applicable, enable automatic recovery, override the default trace properties, or select global variables at runtime.

   For more information, see:

> Note
>
> Setting execution properties here affects a temporary change for the current execution only.

5. Click *OK*.

   As the software begins execution, the execution window opens with the trace log button active.

   Use the buttons at the top of the log window to display the trace log, monitor log, and error log (if there are any errors).

   After the job is complete, use an RDBMS query tool to check the contents of the target table or file.

Related Information

Designer — General [page 200]
Reference Guide: Parameters [page 840]
Reference Guide: Trace properties [page 843]
Setting global variable values [page 408]
Debugging execution errors [page 426]
Examining target data [page 428]

2.14.3.2 Monitor tab

The *Monitor* tab lists the trace logs of all current or most recent executions of a job.

The traffic-light icons in the *Monitor* tab have the following meanings:

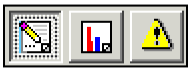
- A green light indicates that the job is running
  You can right-click and select Kill Job to stop a job that is still running.
- A red light indicates that the job has stopped
  You can right-click and select Properties to add a description for a specific trace log. This description is saved with the log which can be accessed later from the Log tab.
- A red cross indicates that the job encountered an error

2.14.3.3 Log tab

You can also select the *Log* tab to view a job's trace log history.

Click a trace log to open it in the workspace.

Use the trace, monitor, and error log icons (left to right at the top of the job execution window in the workspace) to view each type of available log for the date and time that the job was run.



2.14.4 Debugging execution errors

The following tables lists tools that can help you understand execution errors:

Tool	Definition
Trace log	Itemizes the steps executed in the job and the time execution began and ended.
Monitor log	Displays each step of each data flow in the job, the number of rows streamed through each step, and the duration of each step.
Error log	Displays the name of the object being executed when an error occurred and the text of the resulting error message. If the job ran against SAP data, some of the ABAP errors are also available in the error log.
Target data	Always examine your target data to see if your job produced the results you expected.

Related Information

Using logs [page 426]
Examining trace logs [page 428]
Examining monitor logs [page 428]
Examining error logs [page 428]
Examining target data [page 428]

2.14.4.1 Using logs

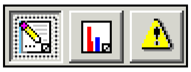
This section describes how to use logs in the Designer.

- To open the trace log on job execution, select ▶ *Tools* › *Options* › *Designer* › *General* › *Open monitor on job execution* ◀.
- To copy log content from an open log, select one or multiple lines and use the key commands [Ctrl+C].

2.14.4.1.1 To access a log during job execution

If your Designer is running when job execution begins, the execution window opens automatically, displaying the trace log information.

Use the monitor and error log icons (middle and right icons at the top of the execution window) to view these logs.



The execution window stays open until you close it.

2.14.4.1.2 To access a log after the execution window has been closed

1. In the project area, click the *Log* tab.
2. Click a job name to view all trace, monitor, and error log files in the workspace. Or expand the job you are interested in to view the list of trace log files and click one.

   Log indicators signify the following:

Job Log Indicator	Description
✓ N_	Indicates that the job executed successfully on this explicitly selected Job Server.
🖧	Indicates that the was job executed successfully by a server group. The Job Server listed executed the job.
☒	Indicates that the job encountered an error on this explicitly selected Job Server.
🖧✗	Indicates that the job encountered an error while being executed by a server group. The Job Server listed executed the job.

3. Click the log icon for the execution of the job you are interested in. (Identify the execution from the position in sequence or datetime stamp.)
4. Use the list box to switch between log types or to view *No logs* or *All logs*.

2.14.4.1.3 To delete a log

You can set how long to keep logs in Administrator.

If want to delete logs from the Designer manually:

1. In the project area, click the *Log* tab.
2. Right-click the log you want to delete and select *Delete Log*.

Related Information

Administrator Guide: Setting the log retention period [page 57]

2.14.4.1.4 Examining trace logs

Use the trace logs to determine where an execution failed, whether the execution steps occur in the order you expect, and which parts of the execution are the most time consuming.

2.14.4.1.5 Examining monitor logs

The monitor log quantifies the activities of the components of the job. It lists the time spent in a given component of a job and the number of data rows that streamed through the component.

2.14.4.1.6 Examining error logs

The software produces an error log for every job execution. Use the error logs to determine how an execution failed. If the execution completed without error, the error log is blank.

2.14.4.2 Examining target data

The best measure of the success of a job is the state of the target data. Always examine your data to make sure the data movement operation produced the results you expect. Be sure that:

- Data was not converted to incompatible types or truncated.
- Data was not duplicated in the target.
- Data was not lost between updates of the target.
- Generated keys have been properly incremented.
- Updated values were handled properly.

2.14.5 Changing Job Server options

Familiarize yourself with the more technical aspects of how the software handles data (using the *Reference Guide*) and some of its interfaces like those for adapters and SAP application.

There are many options available in the software for troubleshooting and tuning a job.

Option	Option description	Default value
Adapter Data Exchange Time-out	(For adapters) Defines the time a function call or outbound message will wait for the response from the adapter operation.	10800000 (3 hours)

Option	Option description	Default value
Adapter Start Time-out	(For adapters) Defines the time that the Administrator or Designer will wait for a response from the Job Server that manages adapters (start/stop/status).	90000 (90 seconds)
AL_JobServerLoadBalanceDebug	Enables a Job Server to log server group information if the value is set to `TRUE`. Information is saved in: **`<$LINK_DIR>`** `/log/`**`<JobServerName>`**`/server_eventlog.txt`.	FALSE
AL_JobServerLoadOSPolling	Sets the polling interval (in seconds) that the software uses to get status information used to calculate the load balancing index. This index is used by server groups.	60
Display DI Internal Jobs	Displays the software's internal datastore CD_DS_d0cafae2 and its related jobs in the object library. The CD_DS_d0cafae2 datastore supports two internal jobs. The first calculates usage dependencies on repository tables and the second updates server group configurations. If you change your repository password, user name, or other connection information, change the default value of this option to TRUE, close and reopen the Designer, then update the CD_DS_d0cafae2 datastore configuration to match your new repository configuration. This enables the calculate usage dependency job (CD_JOBd0cafae2) and the server group job (di_job_al_mach_info) to run without a connection error.	FALSE
FTP Number of Retry	Sets the number of retries for an FTP connection that initially fails.	0
FTP Retry Interval	Sets the FTP connection retry interval in milliseconds.	1000
Global_DOP	Sets the Degree of Parallelism for all data flows run by a given Job Server. You can also set the *Degree of parallelism* for individual data flows from each data flow's Properties window. If a data flow's *Degree of parallelism* value is `0`, then the Job Server will use the Global_DOP value. The Job Server will use the data flow's *Degree of parallelism* value if it is set to any value except zero because it overrides the Global_DOP value.	2
Ignore Reduced Msg Type	(For SAP applications) Disables IDoc reduced message type processing for all message types if the value is set to `TRUE`.	FALSE

Option	Option description	Default value
Ignore Reduced Msg Type_foo	(For SAP application) Disables IDoc reduced message type processing for a specific message type (such as `<foo>`) if the value is set to `TRUE`.	FALSE
OCI Server Attach Retry	The engine calls the Oracle `OCIServerAttach` function each time it makes a connection to Oracle. If the engine calls this function too fast (processing parallel data flows for example), the function may fail. To correct this, increase the retry value to `5`.	3
Splitter Optimization	The software might hang if you create a job in which a file source feeds into two queries. If this option is set to TRUE, the engine internally creates two source files that feed the two queries instead of a splitter that feeds the two queries.	FALSE
Use Explicit Database Links	Jobs with imported database links normally will show improved performance because the software uses these links to push down processing to a database. If you set this option to FALSE, all data flows will not use linked datastores. The use of linked datastores can also be disabled from any data flow properties dialog. The data flow level option takes precedence over this Job Server level option.	TRUE
Use Domain Name	Adds a domain name to a Job Server name in the repository. This creates a fully qualified server name and allows the Designer to locate a Job Server on a different domain. If the fully qualified host name (including domain) is not stored properly, the connection between the Designer or Job Server and the repository may fail. If this problem occurs, change this option to FALSE before adding repositories.	TRUE

Related Information

Performance Optimization Guide: Using parallel Execution, Degree of parallelism [page 2142]

Performance Optimization Guide: Maximizing Push-Down Operations, Database link support for push-down operations across datastores [page 2124]

2.14.5.1 To change option values for an individual Job Server

1. Select the Job Server you want to work with by making it your default Job Server.
   a) Select *Tools* > *Options* > *Designer* > *Environment* .
   b) Select a Job Server from the *Default Job Server* section.
   c) Click *OK*.
2. Select *Tools* > *Options* > *Job Server* > *General* .
3. Enter the section and key you want to use from the following list of value pairs:

Section	Key
`int`	`AdapterDataExchangeTimeout`
`int`	`AdapterStartTimeout`
`AL_JobServer`	`AL_JobServerLoadBalanceDebug`
`AL_JobServer`	`AL_JobServerLoadOSPolling`
`string`	`DisplayDIInternalJobs`
`AL_Engine`	`FTPNumberOfRetry`
`AL_Engine`	`FTPRetryInterval`
`AL_Engine`	`Global_DOP`
`AL_Engine`	`IgnoreReducedMsgType`
`AL_Engine`	`IgnoreReducedMsgType_foo`
`AL_Engine`	`OCIServerAttach_Retry`
`AL_Engine`	`SPLITTER_OPTIMIZATION`
`AL_Engine`	`UseExplicitDatabaseLinks`
`Repository`	`UseDomainName`

4. Enter a value.

   For example, enter the following to change the default value for the number of times a Job Server will retry to make an FTP connection if it initially fails:

Option	Sample value
Section	AL_Engine
Key	FTPNumberOfRetry
Value	2

These settings will change the default value for the FTPNumberOfRetry option from zero to two.

5. To save the settings and close the Options window, click *OK*.
6. Re-select a default Job Server by repeating step 1, as needed.

2.14.5.2 To use mapped drive names in a path

The software supports only UNC (Universal Naming Convention) paths to directories. If you set up a path to a mapped drive, the software will convert that mapped drive to its UNC equivalent.

To make sure that your mapped drive is not converted back to the UNC path, you need to add your drive names in the *Options* window in the Designer.

1. Choose *Tools* > *Options* .
2. In the *Options* window, expand *Job Server* and then select *General*.
3. In the *Section* edit box, enter **`MappedNetworkDrives`**.
4. In the *Key* edit box, enter **`LocalDrive1`** to map to a local drive or **`RemoteDrive1`** to map to a remote drive.
5. In the *Value* edit box, enter a drive letter, such as **`M:\`** for a local drive or **`\\<machine_name>\<share_name>`** for a remote drive.
6. Click OK to close the window.

If you want to add another mapped drive, you need to close the *Options* window and re-enter. Be sure that each entry in the *Key* edit box is a unique name.

2.15 Data Assessment

With operational systems frequently changing, data quality control becomes critical in your extract, transform and load (ETL) jobs. The Designer provides data quality controls that act as a firewall to identify and fix errors in your data. These features can help ensure that you have trusted information.

The Designer provides the following features that you can use to determine and improve the quality and structure of your source data:

- Use the Data Profiler to determine:
  - The quality of your source data before you extract it. The Data Profiler can identify anomalies in your source data to help you better define corrective actions in the Validation transform, data quality, or other transforms.
  - The distribution, relationship, and structure of your source data to better design your jobs and data flows, as well as your target data warehouse.

  - The content of your source and target data so that you can verify that your data extraction job returns the results you expect.
- Use the View Data feature to:
  - View your source data before you execute a job to help you create higher quality job designs.
  - Compare sample data from different steps of your job to verify that your data extraction job returns the results you expect.
- Use the Design-Time Data Viewer feature to view and analyze the input and output for a data set in real time as you design a transform even before data flow is complete or valid.
- Use the Validation transform to:
  - Verify that your source data meets your business rules.
  - Take appropriate actions when the data does not meet your business rules.
- Use the auditing data flow feature to:
  - Define rules that determine if a source, transform, or target object processes correct data.
  - Define the actions to take when an audit rule fails.
- Use data quality transforms to improve the quality of your data.
- Use Data Validation dashboards in the Metadata Reporting tool to evaluate the reliability of your target data based on the validation rules you created in your batch jobs. This feedback allows business users to quickly review, assess, and identify potential inconsistencies or errors in source data.

Related Information

Using the Data Profiler [page 433]
Using View Data to determine data quality [page 448]
Using the Design-Time Data Viewer [page 699]
Using the Validation transform [page 450]
Using Auditing [page 452]
Overview of data quality [page 464]
Management Console Guide: Data Validation Dashboard Reports [page 1981]

2.15.1 Using the Data Profiler

The Data Profiler executes on a profiler server to provide the following data profiler information that multiple users can view:

- Column analysis—The Data Profiler provides two types of column profiles:
  - Basic profiling—This information includes minimum value, maximum value, average value, minimum string length, and maximum string length.
  - Detailed profiling—Detailed column analysis includes distinct count, distinct percent, median, median string length, pattern count, and pattern percent.
- Relationship analysis—This information identifies data mismatches between any two columns for which you define a relationship, including columns that have an existing primary key and foreign key relationship. You can save two levels of data:

- Save the data only in the columns that you select for the relationship.
- Save the values in all columns in each row.

2.15.1.1 Data sources that you can profile

You can execute the Data Profiler on data contained in the following sources. See the *Release Notes* for the complete list of sources that the Data Profiler supports.

- Databases, which include:
  - Attunity Connector for mainframe databases
  - DB2
  - Oracle
  - SQL Server
  - SAP Sybase IQ
  - SAP Sybase SQL Anywhere
  - Teradata
- Applications, which include:
  - JDE One World
  - JDE World
  - Oracle Applications
  - PeopleSoft
  - SAP Applications
  - SAP Master Data Services
  - SAP NetWeaver Business Warehouse
  - Siebel
- Flat files

2.15.1.2 Connecting to the profiler server

You must install and configure the profiler server before you can use the Data Profiler.

The Designer must connect to the profiler server to run the Data Profiler and view the profiler results. You provide this connection information on the Profiler Server Login window.

1. Use one of the following methods to invoke the Profiler Server Login window:
   - From the tool bar menu, select *Tools* > *Profiler Server Login* .
   - On the bottom status bar, double-click the Profiler Server icon which is to the right of the Job Server icon.
2. Enter your user credentials for the CMS.
   - *System*—Specify the server name and optionally the port for the CMS.
   - *User name*—Specify the user name to use to log into CMS.
   - *Password*—Specify the password to use to log into the CMS.
   - *Authentication*—Specify the authentication type used by the CMS.

3. Click *Log on*.
   The software attempts to connect to the CMS using the specified information. When you log in successfully, the list of profiler repositories that are available to you is displayed.
4. Select the repository you want to use.
5. Click *OK* to connect using the selected repository.

   When you successfully connect to the profiler server, the Profiler Server icon on the bottom status bar no longer has the red X on it. In addition, when you move the pointer over this icon, the status bar displays the location of the profiler server.

Related Information

Management Console Guide: Profile Server Management [page 1929]
Administrator Guide: User and rights management [page 46]

2.15.1.3 Profiler statistics

2.15.1.3.1 Column profile

You can generate statistics for one or more columns. The columns can all belong to one data source or from multiple data sources. If you generate statistics for multiple sources in one profile task, all sources must be in the same datastore.

Basic profiling

By default, the Data Profiler generates the following basic profiler attributes for each column that you select.

Basic attribute	Description
Min	Of all values, the lowest value in this column.
Min count	Number of rows that contain this lowest value in this column.
Max	Of all values, the highest value in this column.
Max count	Number of rows that contain this highest value in this column.
Average	For numeric columns, the average value in this column.
Min string length	For character columns, the length of the shortest string value in this column.
Max string length	For character columns, the length of the longest string value in this column.
Average string length	For character columns, the average length of the string values in this column.
Nulls	Number of `NULL` values in this column.

Basic attribute	Description
Nulls %	Percentage of rows that contain a `NULL` value in this column.
Zeros	Number of `0` values in this column.
Zeros %	Percentage of rows that contain a `0` value in this column.
Blanks	For character columns, the number of rows that contain a blank in this column.
Blanks %	Percentage of rows that contain a blank in this column.

Detailed profiling

You can generate more detailed attributes in addition to the above attributes, but detailed attributes generation consumes more time and computer resources. Therefore, it is recommended that you do not select the detailed profile unless you need the following attributes:

Detailed attribute	Description
Median	The value that is in the middle row of the source table.
Median string length	For character columns, the value that is in the middle row of the source table.
Distincts	Number of distinct values in this column.
Distinct %	Percentage of rows that contain each distinct value in this column.
Patterns	Number of different patterns in this column.
Pattern %	Percentage of rows that contain each pattern in this column.

Examples of using column profile statistics to improve data quality

You can use the column profile attributes to assist you in different tasks, including the following tasks:

- Obtain basic statistics, frequencies, ranges, and outliers. For example, these profile statistics might show that a column value is markedly higher than the other values in a data source. You might then decide to define a validation transform to set a flag in a different table when you load this outlier into the target table.
- Identify variations of the same content. For example, part number might be an integer data type in one data source and a varchar data type in another data source. You might then decide which data type you want to use in your target data warehouse.
- Discover data patterns and formats. For example, the profile statistics might show that phone number has several different formats. With this profile information, you might decide to define a validation transform to convert them all to use the same target format.
- Analyze the numeric range. For example, customer number might have one range of numbers in one source, and a different range in another source. Your target will need to have a data type that can accommodate the maximum range.
- Identify missing information, nulls, and blanks in the source system. For example, the profile statistics might show that nulls occur for fax number. You might then decide to define a validation transform to replace the null value with a phrase such as "Unknown" in the target table.

Related Information

To view the column attributes generated by the Data Profiler [page 444]
Submitting column profiler tasks [page 438]

2.15.1.3.2 Relationship profile

A relationship profile shows the percentage of non matching values in columns of two sources. The sources can be:

- Tables
- Flat files
- A combination of a table and a flat file

The key columns can have a primary key and foreign key relationship defined or they can be unrelated (as when one comes from a datastore and the other from a file format).

You can choose between two levels of relationship profiles to save:

- *Save key columns data only*—By default, the Data Profiler saves the data only in the columns that you select for the relationship.

  > Note
  >
  > The Save key columns data only level is not available when using Oracle datastores.

- *Save all columns data*—You can save the values in the other columns in each row, but this processing will take longer and consume more computer resources to complete.

When you view the relationship profile results, you can drill down to see the actual data that does not match.

You can use the relationship profile to assist you in different tasks, including the following tasks:

- Identify missing data in the source system. For example, one data source might include region, but another source might not.
- Identify redundant data across data sources. For example, duplicate names and addresses might exist between two sources or no name might exist for an address in one source.
- Validate relationships across data sources. For example, two different problem tracking systems might include a subset of common customer-reported problems, but some problems only exist in one system or the other.

Related Information

Submitting relationship profiler tasks [page 440]
Viewing the profiler results [page 443]

2.15.1.4 Executing a profiler task

The Data Profiler allows you to calculate profiler statistics for any set of columns you choose.

> **Note**
>
> This optional feature is not available for columns with nested schemas, LONG or TEXT data type.

You cannot execute a column profile task with a relationship profile task.

2.15.1.4.1 Submitting column profiler tasks

1. In the Object Library of the Designer, you can select either a table or flat file.

   For a table, go to the *Datastores* tab and select a table. If you want to profile all tables within a datastore, select the datastore name. To select a subset of tables in the *Ddatastore* tab, hold down the Ctrl key as you select each table.

   For a flat file, go to the *Formats* tab and select a file.
2. After you select your data source, you can generate column profile statistics in one of the following ways:
   - Right-click and select *Submit Column Profile Request*.
     Some of the profile statistics can take a long time to calculate. Select this method so the profile task runs asynchronously and you can perform other Designer tasks while the profile task executes.
     This method also allows you to profile multiple sources in one profile task.
   - Right-click, select *View Data*, click the *Profile* tab, and click *Update*. This option submits a synchronous profile task, and you must wait for the task to complete before you can perform other tasks in the Designer .
     You might want to use this option if you are already in the *View Data* window and you notice that either the profile statistics have not yet been generated, or the date that the profile statistics were generated is older than you want.
3. (Optional) Edit the profiler task name.

   The Data Profiler generates a default name for each profiler task. You can edit the task name to create a more meaningful name, a unique name, or to remove dashes which are allowed in column names but not in task names.

   If you select a single source, the default name has the following format:

   `<username_t_sourcename>`

   If you select multiple sources, the default name has the following format:

   `<username_t_firstsourcename_lastsourcename>`

Column	Description
username	Name of the user that the software uses to access system services.
t	Type of profile. The value is `C` for column profile that obtains attributes (such as low value and high value) for each selected column.

Column	Description
firstsourcename	Name of first source in alphabetic order.
lastsourcename	Name of last source in alphabetic order if you select multiple sources.

4. If you select one source, the *Submit Column Profile Request* window lists the columns and data types.

   Keep the check in front of each column that you want to profile and remove the check in front of each column that you do not want to profile.

   Alternatively, you can click the check box at the top in front of *Name* to deselect all columns and then select the check boxes.
5. If you selected multiple sources, the *Submit Column Profiler Request* window lists the sources on the left.
   a) Select a data source to display its columns on the right side.
   b) On the right side of the *Submit Column Profile Request* window, keep the check in front of each column that you want to profile, and remove the check in front of each column that you do not want to profile.

      Alternatively, you can click the check box at the top in front of Name to deselect all columns and then select the individual check box for the columns you want to profile.
   c) Repeat steps 1 and 2 for each data source.
6. (Optional) Select *Detailed profiling* for a column.

   > i Note
   >
   > The Data Profiler consumes a large amount of resources when it generates detailed profile statistics. Choose Detailed profiling only if you want these attributes: distinct count, distinct percent, median value, median string length, pattern, pattern count. If you choose Detailed profiling, ensure that you specify a pageable cache directory that contains enough disk space for the amount of data you profile.

   If you want detailed attributes for all columns in all sources listed, click *Detailed profiling* and select *Apply* to all columns of all sources.

   If you want to remove Detailed profiling for all columns, click *Detailed profiling* and select *Remove* from all columns of all sources.
7. Click *Submit* to execute the profile task.

   > i Note
   >
   > If the table metadata changed since you imported it (for example, a new column was added), you must re-import the source table before you execute the profile task.

   If you clicked the *Submit Column Profile Request* option to reach this *Submit Column Profiler Request* window, the Profiler monitor pane appears automatically when you click *Submit*.

   If you clicked *Update* on the *Profile* tab of the *View Data* window, the *Profiler* monitor window does not appear when you click *Submit*. Instead, a profile task is submitted asynchronously and you must wait for it to complete before you can do other tasks in the Designer .

   You can also monitor your profiler task by name in the Administrator.
8. When the profiler task has completed, you can view the profile results in the *View Data* option.

Related Information

Column profile [page 435]
Monitoring profiler tasks using the Designer [page 442]
Viewing the profiler results [page 443]
Administrator Guide: To configure run-time resources [page 87]
Management Console Guide: Monitoring profiler tasks using the Administrator [page 1933]

2.15.1.4.2 Submitting relationship profiler tasks

A relationship profile shows the percentage of non matching values in columns of two sources. The sources can be any of the following:

- Tables
- Flat files
- A combination of a table and a flat file

The columns can have a primary key and foreign key relationship defined or they can be unrelated (as when one comes from a datastore and the other from a file format).

The two columns do not need to be the same data type, but they must be convertible. For example, if you run a relationship profiler task on an integer column and a varchar column, the Data Profiler converts the integer value to a varchar value to make the comparison.

> i Note
>
> The Data Profiler consumes a large amount of resources when it generates relationship values. If you plan to use Relationship profiling, ensure that you specify a pageable cache directory that contains enough disk space for the amount of data you profile.

Related Information

Data sources that you can profile [page 434]
Administrator Guide: To configure run-time resources [page 87]

2.15.1.4.2.1 To generate a relationship profile for columns in two sources

1. In the Object Library of the Designer, select two sources.

   To select two sources in the same datastore or file format:

   a) Go to the *Datastore* or *Format* tab in the Object Library.

b) Hold the Ctrl key down as you select the second table.
c) Right-click and select *Submit Relationship Profile Request*.

To select two sources from different datastores or files:

a) Go to the *Datastore* or *Format* tab in the Object Library.
b) Right-click on the first source, select *Submit Relationship Profile Request* > *Relationship with* .
c) Change to a different Datastore or Format in the Object Library
d) Click on the second source.

The *Submit Relationship Profile Request* window appears.

> **Note**
>
> You cannot create a relationship profile for columns with a LONG or TEXT data type.

2. (Optional) Edit the profiler task name.

   You can edit the task name to create a more meaningful name, a unique name, or to remove dashes, which are allowed in column names but not in task names. The default name that the Data Profiler generates for multiple sources has the following format:

   ```
   <username>_<t>_<firstsourcename>_<lastsourcename>
   ```

Column	Description
username	Name of the user that the software uses to access system services.
t	Type of profile. The value is `R` for Relationship profile that obtains non matching values in the two selected columns.
firstsourcename	Name first selected source.
lastsourcename	Name last selected source.

3. By default, the upper pane of the *Submit Relationship Profile Request* window shows a line between the primary key column and foreign key column of the two sources, if the relationship exists. You can change the columns to profile.

   The bottom half of the *Submit Relationship Profile Request* window shows that the profile task will use the equal (=) operation to compare the two columns. The Data Profiler will determine which values are not equal and calculate the percentage of non matching values.
4. To delete an existing relationship between two columns, select the line, right-click, and select *Delete Selected Relation*.

   To delete all existing relationships between the two sources, do one of the following actions:

   ○ Right-click in the upper pane and click *Delete All Relations*.
   ○ Click *Delete All Relations* near the bottom of the *Submit Relationship Profile Request* window.
5. If a primary key and foreign key relationship does not exist between the two data sources, specify the columns to profile. You can resize each data source to show all columns.

   To specify or change the columns for which you want to see relationship values:

   a) Move the cursor to the first column to select. Hold down the cursor and draw a line to the other column that you want to select.

b) If you deleted all relations and you want the Data Profiler to select an existing primary-key and foreign-key relationship, either right-click in the upper pane and click *Propose Relation*, or click *Propose Relation* near the bottom of the *Submit Relationship Profile Request* window.

6. By default, the *Save key columns data only* option is selected. This option indicates that the Data Profiler saves the data only in the columns that you select for the relationship, and you will not see any sample data in the other columns when you view the relationship profile.

   If you want to see values in the other columns in the relationship profile, select the *Save all columns data* option.
7. Click *Submit* to execute the profiler task.

   > i Note
   >
   > If the table metadata changed since you imported it (for example, a new column was added), you must re-import the source table before you execute the profile task.
8. The Profiler monitor pane appears automatically when you click *Submit*.

   You can also monitor your profiler task by name in the Administrator.
9. When the profiler task has completed, you can view the profile results in the *View Data* option when you right click on a table in the Object Library.

Related Information

To view the relationship profile data generated by the Data Profiler [page 446]
Monitoring profiler tasks using the Designer [page 442]
Management Console Guide: Monitoring profiler tasks using the Administrator [page 1933]
Viewing the profiler results [page 443]

2.15.1.5 Monitoring profiler tasks using the Designer

The *Profiler* monitor window appears automatically when you submit a profiler task if you clicked the menu bar to view the *Profiler* monitor window. You can dock this profiler monitor pane in the Designer or keep it separate.

The Profiler monitor pane displays the currently running task and all of the profiler tasks that have executed within a configured number of days.

The icons in the upper-left corner of the Profiler monitor display the following information:

- Click the *Refresh* icon to refresh the Profiler monitor pane and display the latest status of profiler tasks.
- Click the *Information* icon to view the sources that the selected task is profiling. If the task failed, the Information window also displays an error message.

The Profiler monitor shows the following columns:

Column	Description

Name	Name of the profiler task that was submitted from the Designer.
	If the profiler task is for a single source, the default name has the following format:
	`<username_t_sourcename>`
	If the profiler task is for multiple sources, the default name has the following format:
	`<username_t_firstsourcename_lastsourcename>`

Type	The type of profiler task can be: ● Column ● Relationship
Status	The status of a profiler task can be: ● Done—The task completed successfully. ● Pending—The task is on the wait queue because the maximum number of concurrent tasks has been reached or another task is profiling the same table. ● Running—The task is currently executing. ● Error—The task terminated with an error. Double-click on the value in this Status column to display the error message.
Timestamp	Date and time that the profiler task executed.
Sources	Names of the tables for which the profiler task executes.

Related Information

Executing a profiler task [page 438]
Management Console Guide: Configuring profiler task parameters [page 1930]

2.15.1.6 Viewing the profiler results

The Data Profiler calculates and saves the profiler attributes into a profiler repository that multiple users can view.

Related Information

To view the column attributes generated by the Data Profiler [page 444]
To view the relationship profile data generated by the Data Profiler [page 446]

2.15.1.6.1 To view the column attributes generated by the Data Profiler

1. In the Object Library, select the table for which you want to view profiler attributes.
2. Right-click and select *View Data*.
3. Click the *Profile* tab (second) to view the column profile attributes.
   a) The *Profile* tab shows the number of physical records that the Data Profiler processed to generate the values in the profile grid.
   b) The profile grid contains the column names in the current source and profile attributes for each column. To populate the profile grid, execute a profiler task or select names from this column and click *Update*.
   c) You can sort the values in each attribute column by clicking the column heading. The value `n/a` in the profile grid indicates an attribute does not apply to a data type,

Basic profile attribute	Description	Relevant data type		
		Character	Numeric	Datetime
Min	Of all values, the lowest value in this column.	Yes	Yes	Yes
Min count	Number of rows that contain this lowest value in this column.	Yes	Yes	Yes
Max	Of all values, the highest value in this column.	Yes	Yes	Yes
Max count	Number of rows that contain this highest value in this column.	Yes	Yes	Yes
Average	For numeric columns, the average value in this column.	n/a	Yes	n/a
Min string length	For character columns, the length of the shortest string value in this column.	Yes	No	No
Max string length	For character columns, the length of the longest string value in this column.	Yes	No	No
Average string length	For character columns, the average length of the string values in this column.	Yes	No	No
Nulls	Number of `NULL` values in this column.	Yes	Yes	Yes
Nulls %	Percentage of rows that contain a `NULL` value in this column.	Yes	Yes	Yes
Zeros	Number of `0` values in this column.	No	Yes	No
Zeros %	Percentage of rows that contain a `0` value in this column.	No	Yes	No
Blanks	For character columns, the number of rows that contain a blank in this column.	Yes	No	No
Blanks %	Percentage of rows that contain a blank in this column.	Yes	No	No

   d) If you selected the *Detailed profiling* option on the *Submit Column Profile Request* window, the *Profile* tab also displays the following detailed attribute columns.

Detailed profile attribute	Description	Relevant data type		
		Character	Numeric	Datetime
Distincts	Number of distinct values in this column.	Yes	Yes	Yes
Distinct %	Percentage of rows that contain each distinct value in this column.	Yes	Yes	Yes
Median	The value that is in the middle row of the source table.	Yes	Yes	Yes
Median string length	For character columns, the value that is in the middle row of the source table.	Yes	No	No
Pattern %	Percentage of rows that contain each distinct value in this column. The format of each unique pattern in this column.	Yes	No	No
Patterns	Number of different patterns in this column.	Yes	No	No

4. Click an attribute value to view the entire row in the source table. The bottom half of the *View Data* window displays the rows that contain the attribute value that you clicked. You can hide columns that you do not want to view by clicking the Show/Hide Columns icon.

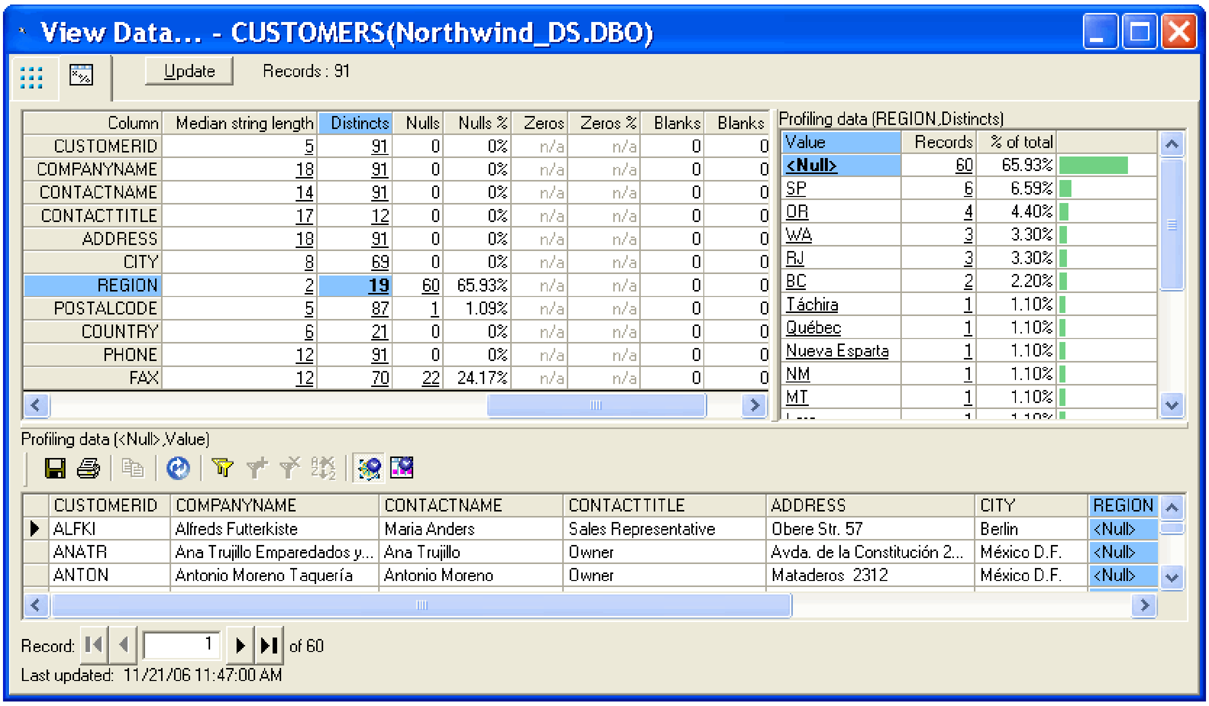
   For example, your target ADDRESS column might only be 45 characters, but the Profiling data for this Customer source table shows that the maximum string length is 46. Click the value 46 to view the actual data. You can resize the width of the column to display the entire string.
5. (Optional) Click *Update* if you want to update the profile attributes. Reasons to update at this point include:
   - The profile attributes have not yet been generated.
   - The date that the profile attributes were generated is older than you want. The Last updated value in the bottom left corner of the *Profile* tab is the timestamp when the profile attributes were last generated.

   > Note
   >
   > The Update option submits a synchronous profiler task, and you must wait for the task to complete before you can perform other tasks in the Designer.

   The *Submit column Profile Request* window appears.

   Select only the column names you need for this profiling operation because Update calculations impact performance. You can also click the check box at the top in front of Name to deselect all columns and then select each check box in front of each column you want to profile.
6. Click a statistic in either Distincts or Patterns to display the percentage of each distinct value or pattern value in a column. The pattern values, number of records for each pattern value, and percentages appear on the right side of the *Profile* tab.

   For example, the following *Profile* tab for table CUSTOMERS shows the profile attributes for column REGION. The Distincts attribute for the REGION column shows the statistic 19 which means 19 distinct values for REGION exist.



View Data... - CUSTOMERS(Northwind_DS.DBO)

Records : 91

Column	Median string length	Distincts	Nulls	Nulls %	Zeros	Zeros %	Blanks	Blanks
CUSTOMERID	5	91	0	0%	n/a	n/a	0	0
COMPANYNAME	18	91	0	0%	n/a	n/a	0	0
CONTACTNAME	14	91	0	0%	n/a	n/a	0	0
CONTACTTITLE	17	12	0	0%	n/a	n/a	0	0
ADDRESS	18	91	0	0%	n/a	n/a	0	0
CITY	8	69	0	0%	n/a	n/a	0	0
REGION	2	19	60	65.93%	n/a	n/a	0	0
POSTALCODE	5	87	1	1.09%	n/a	n/a	0	0
COUNTRY	6	21	0	0%	n/a	n/a	0	0
PHONE	12	91	0	0%	n/a	n/a	0	0
FAX	12	70	22	24.17%	n/a	n/a	0	0

Profiling data (REGION,Distincts)

Value	Records	% of total	
<Null>	60	65.93%	🟩
SP	6	6.59%	🟩
OR	4	4.40%	🟩
WA	3	3.30%	🟩
RJ	3	3.30%	🟩
BC	2	2.20%	🟩
Táchira	1	1.10%	🟩
Québec	1	1.10%	🟩
Nueva Esparta	1	1.10%	🟩
NM	1	1.10%	🟩
MT	1	1.10%	🟩

Profiling data (<Null>,Value)

CUSTOMERID	COMPANYNAME	CONTACTNAME	CONTACTTITLE	ADDRESS	CITY	REGION
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	<Null>
ANATR	Ana Trujillo Emparedados y...	Ana Trujillo	Owner	Avda. de la Constitución 2...	México D.F.	<Null>
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	<Null>

Record: 1 of 60

Last updated: 11/21/06 11:47:00 AM

7. Click the statistic in the Distincts column to display each of the 19 values and the percentage of rows in table CUSTOMERS that have that value for column REGION. In addition, the bars in the right-most column show the relative size of each percentage.
8. The Profiling data on the right side shows that a very large percentage of values for REGION is Null. Click either `Null` under Value or `60` under Records to display the other columns in the rows that have a Null value in the REGION column.
9. Your business rules might dictate that REGION should not contain Null values in your target data warehouse. Therefore, decide what value you want to substitute for Null values when you define a validation transform.

Related Information

Executing a profiler task [page 438]

Defining a validation rule based on a column profile [page 451]

2.15.1.6.2 To view the relationship profile data generated by the Data Profiler

Relationship profile data shows the percentage of non matching values in columns of two sources. The sources can be tables, flat files, or a combination of a table and a flat file. The columns can have a primary key and foreign key relationship defined or they can be unrelated (as when one comes from a datastore and the other from a file format).

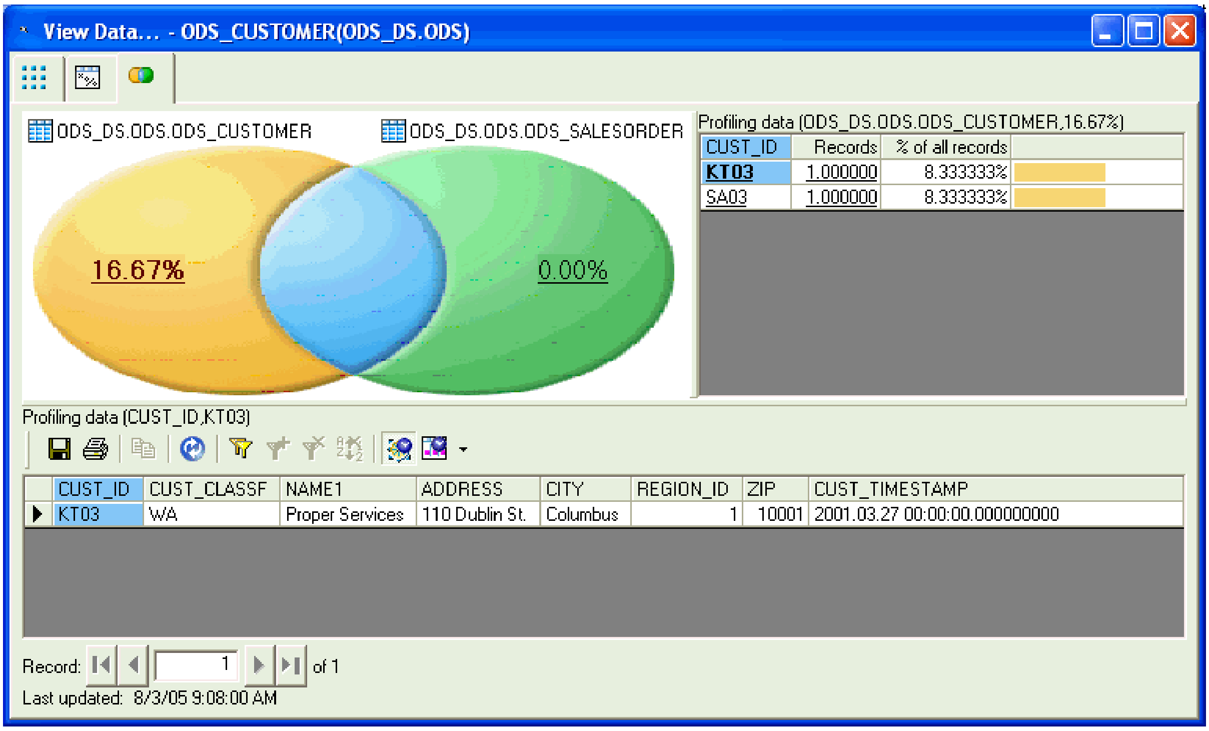
1. In the Object Library, select the table or file for which you want to view relationship profile data.

2. Right-click and select *View Data*.
3. Click the *Relationship* tab (third) to view the relationship profile results.

> i Note
>
> The *Relationship* tab is visible only if you executed a relationship profiler task.

4. Click the nonzero percentage in the diagram to view the key values that are not contained within the other table.

   For example, the following View Data Relationship tab shows the percentage (16.67) of customers that do not have a sales order. The relationship profile was defined on the CUST_ID column in table ODS_CUSTOMER and CUST_ID column in table ODS_SALESORDER. The value in the left oval indicates that 16.67% of rows in table ODS_CUSTOMER have CUST_ID values that do not exist in table ODS_SALESORDER.



View Data... - ODS_CUSTOMER(ODS_DS.ODS)
ODS_DS.ODS.ODS_CUSTOMER
ODS_DS.ODS.ODS_SALESORDER
16.67%
0.00%

Profiling data (ODS_DS.ODS.ODS_CUSTOMER,16.67%)

CUST_ID	Records	% of all records
KT03	1.000000	8.333333%
SA03	1.000000	8.333333%

Profiling data (CUST_ID,KT03)

CUST_ID	CUST_CLASSF	NAME1	ADDRESS	CITY	REGION_ID	ZIP	CUST_TIMESTAMP
KT03	WA	Proper Services	110 Dublin St.	Columbus	1	10001	2001.03.27 00:00:00.000000000

Record: 1 of 1

Last updated: 8/3/05 9:08:00 AM

   Click the 16.67 percentage in the ODS_CUSTOMER oval to display the CUST_ID values that do not exist in the ODS_SALESORDER table. The non matching values KT03 and SA03 display on the right side of the Relationship tab. Each row displays a non matching CUST_ID value, the number of records with that CUST_ID value, and the percentage of total customers with this CUST_ID value.

5. Click one of the values on the right side to display the other columns in the rows that contain that value.

   The bottom half of the *Relationship Profile* tab displays the values in the other columns of the row that has the value KT03 in the column CUST_ID.

> i Note
>
> If you did not select Save all column data on the *Submit Relationship Profile Request* window, you cannot view the data in the other columns.

Related Information

Submitting relationship profiler tasks [page 440]

2.15.2 Using View Data to determine data quality

Use View Data to help you determine the quality of your source and target data. View Data provides the capability to:

- View sample source data before you execute a job to create higher quality job designs.
- Compare sample data from different steps of your job to verify that your data extraction job returns the results you expect.

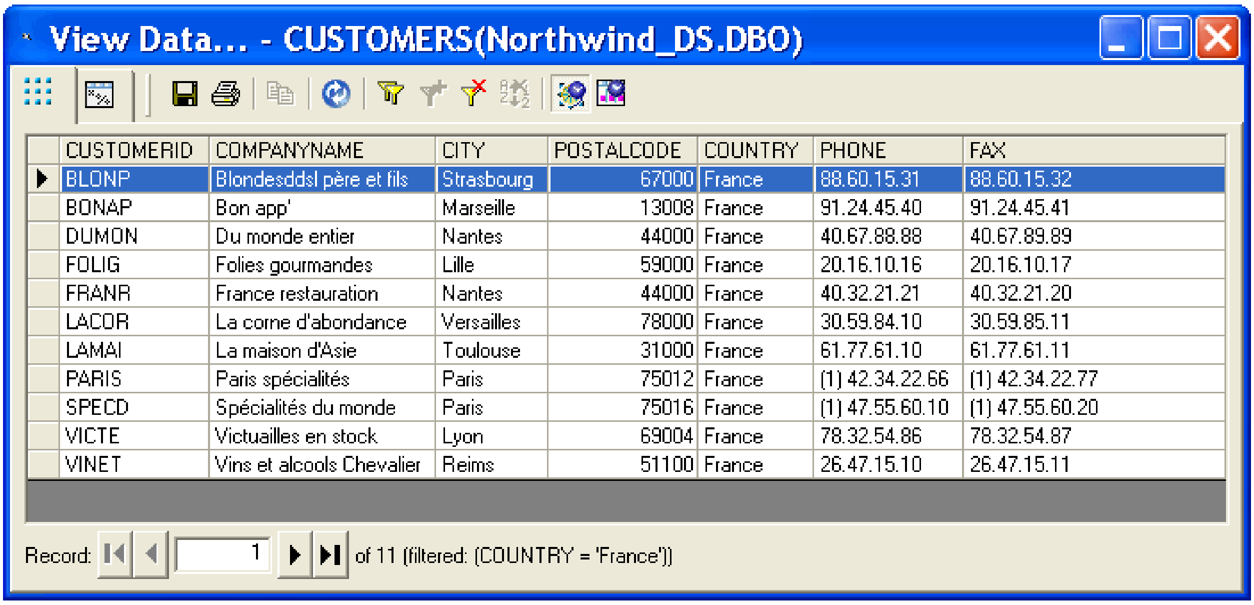
Related Information

Defining a validation rule based on a column profile [page 451]
Using View Data [page 688]

2.15.2.1 Data tab

The *Data* tab is always available and displays the data contents of sample rows. You can display a subset of columns in each row and define filters to display a subset of rows.

For example, your business rules might dictate that all phone and fax numbers be in one format for each country. The following *Data* tab shows a subset of rows for the customers that are in France.



CUSTOMERID	COMPANYNAME	CITY	POSTALCODE	COUNTRY	PHONE	FAX
BLONP	Blondesddsl père et fils	Strasbourg	67000	France	88.60.15.31	88.60.15.32
BONAP	Bon app'	Marseille	13008	France	91.24.45.40	91.24.45.41
DUMON	Du monde entier	Nantes	44000	France	40.67.88.88	40.67.89.89
FOLIG	Folies gourmandes	Lille	59000	France	20.16.10.16	20.16.10.17
FRANR	France restauration	Nantes	44000	France	40.32.21.21	40.32.21.20
LACOR	La corne d'abondance	Versailles	78000	France	30.59.84.10	30.59.85.11
LAMAI	La maison d'Asie	Toulouse	31000	France	61.77.61.10	61.77.61.11
PARIS	Paris spécialités	Paris	75012	France	(1) 42.34.22.66	(1) 42.34.22.77
SPECD	Spécialités du monde	Paris	75016	France	(1) 47.55.60.10	(1) 47.55.60.20
VICTE	Victuailles en stock	Lyon	69004	France	78.32.54.86	78.32.54.87
VINET	Vins et alcools Chevalier	Reims	51100	France	26.47.15.10	26.47.15.11

Record: 1 of 11 (filtered: (COUNTRY = 'France'))

Notice that the PHONE and FAX columns displays values with two different formats. You can now decide which format you want to use in your target data warehouse and define a validation transform accordingly.

Related Information

View Data Properties [page 692]
Defining a validation rule based on a column profile [page 451]
Data tab [page 696]

2.15.2.2 Profile tab

Two displays are available on the *Profile* tab:

- Without the Data Profiler, the *Profile* tab displays the following column attributes: distinct values, NULLs, minimum value, and maximum value.
- If you configured and use the Data Profiler, the *Profile* tab displays the same above column attributes plus many more calculated statistics, such as average value, minimum string length, and maximum string length, distinct count, distinct percent, median, median string length, pattern count, and pattern percent.

Related Information

Profile tab [page 697]
To view the column attributes generated by the Data Profiler [page 444]

2.15.2.3 Relationship Profile or Column Profile tab

The third tab that displays depends on whether or not you configured and use the Data Profiler.

- If you do not use the Data Profiler, the *Column Profile* tab allows you to calculate statistical information for a single column.
- If you use the Data Profiler, the *Relationship* tab displays the data mismatches between two columns from which you can determine the integrity of your data between two sources.

Related Information

Column Profile tab [page 698]
To view the relationship profile data generated by the Data Profiler [page 446]

2.15.3 Using the Validation transform

The Data Profiler and View Data features can identify anomalies in incoming data. You can then use a Validation transform to define the rules that sort good data from bad. You can write the bad data to a table or file for subsequent review.

For details on the Validation transform including how to implement reusable validation functions, see the *SAP Data Services Reference Guide*.

Related Information

Reference Guide: Transforms, Validation [page 1472]

2.15.3.1 Analyzing the column profile

You can obtain column profile information by submitting column profiler tasks.

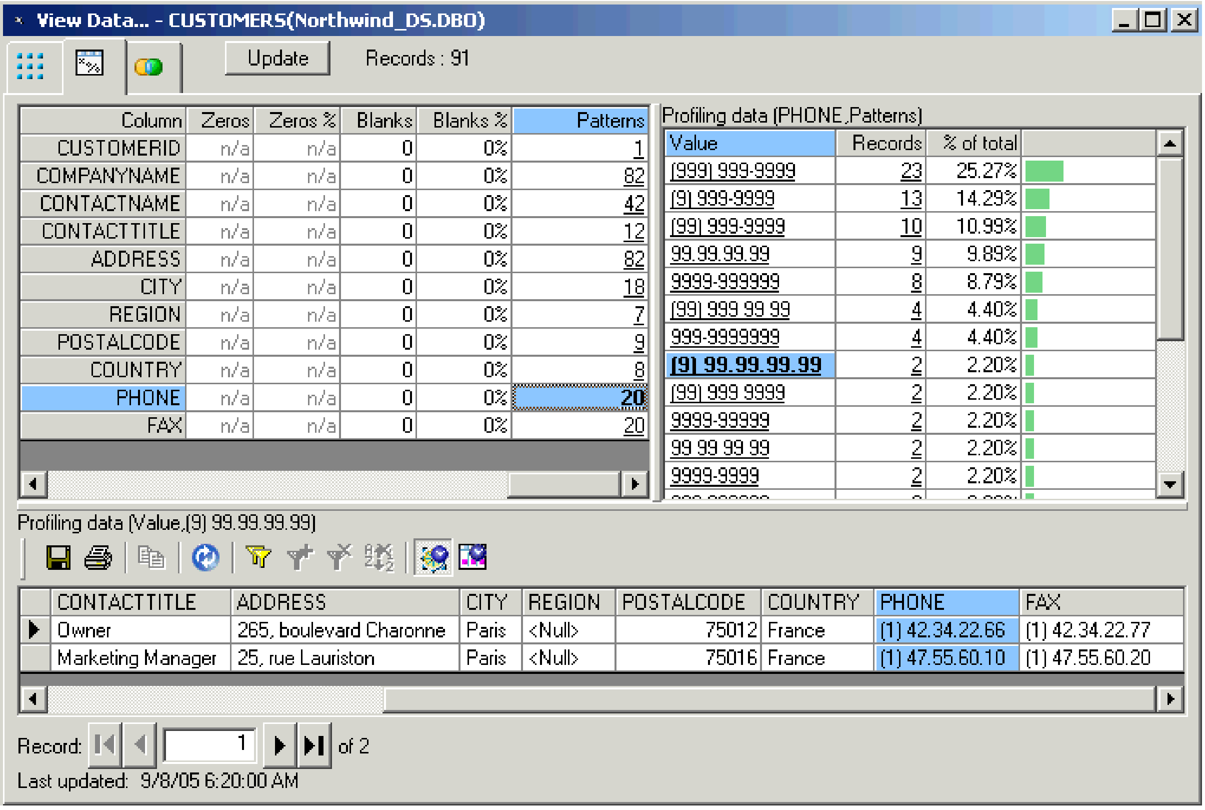
For example, suppose you want to analyze the data in the Customers table in the Microsoft SQL Server Northwinds sample database.

Related Information

Submitting column profiler tasks [page 438]

2.15.3.1.1 To analyze column profile attributes

1. In the object library, right-click the profiled Customers table and select *View Data*.
2. Select the *Profile* tab in the *View Data* window. The Profile tab displays the column-profile attributes shown in the following graphic.



View Data... - CUSTOMERS(Northwind_DS.DBO)

Records : 91

Column	Zeros	Zeros %	Blanks	Blanks %	Patterns
CUSTOMERID	n/a	n/a	0	0%	1
COMPANYNAME	n/a	n/a	0	0%	82
CONTACTNAME	n/a	n/a	0	0%	42
CONTACTTITLE	n/a	n/a	0	0%	12
ADDRESS	n/a	n/a	0	0%	82
CITY	n/a	n/a	0	0%	18
REGION	n/a	n/a	0	0%	7
POSTALCODE	n/a	n/a	0	0%	9
COUNTRY	n/a	n/a	0	0%	8
PHONE	n/a	n/a	0	0%	20
FAX	n/a	n/a	0	0%	20

Profiling data (PHONE,Patterns)

Value	Records	% of total	
(999) 999-9999	23	25.27%	🟩
(9) 999-9999	13	14.29%	🟩
(99) 999-9999	10	10.99%	🟩
99.99.99.99	9	9.89%	🟩
9999-999999	8	8.79%	🟩
(99) 999 99 99	4	4.40%	🟩
999-9999999	4	4.40%	🟩
(9) 99.99.99.99	2	2.20%	🟩
(99) 999 9999	2	2.20%	🟩
9999-99999	2	2.20%	🟩
99 99 99 99	2	2.20%	🟩
9999-9999	2	2.20%	🟩

Profiling data (Value,(9) 99.99.99.99)

CONTACTTITLE	ADDRESS	CITY	REGION	POSTALCODE	COUNTRY	PHONE	FAX
Owner	265, boulevard Charonne	Paris	<Null>	75012	France	(1) 42.34.22.66	(1) 42.34.22.77
Marketing Manager	25, rue Lauriston	Paris	<Null>	75016	France	(1) 47.55.60.10	(1) 47.55.60.20

Record: 1 of 2

Last updated: 9/8/05 6:20:00 AM

The Patterns attribute for the PHONE column shows the value 20, which means 20 different patterns exist.

3. Click the value 20 in the *Patterns* attribute column. The *Profiling data* pane displays the individual patterns for the column PHONE and the percentage of rows for each pattern.
4. Suppose that your business rules dictate that all phone numbers in France should have the format 99.99.99.99. However, the profiling data shows that two records have the format (9) 99.99.99.99. To display the columns for these two records in the bottom pane, click either `(9) 99.99.99.99` under Value or click `2` under Records. You can see that some phone numbers in France have a prefix of (1).

You can use a Validation transform to identify rows containing the unwanted prefix. Then you can correct the data to conform to your busness rules then reload it.

The next section describes how to configure the Validation transform to identify the errant rows.

Related Information

Defining a validation rule based on a column profile [page 451]

2.15.3.2 Defining a validation rule based on a column profile

This section takes the Data Profiler results and defines the Validation transform according to the sample business rules. Based on the preceding example of the phone prefix (1) for phone numbers in France, the following procedure describes how to define a data flow and validation rule that identifies that pattern. You can then review the failed data, make corrections, and reload the data.

2.15.3.2.1 To define the validation rule that identifies a pattern

This procedure describes how to define a data flow and validation rule that identifies rows containing the (1) prefix described in the previous section.

1. Create a data flow with the Customers table as a source, add a Validation transform and a target, and connect the objects.
2. Open the Validation transform by clicking its name.
3. In the transform editor, click *Add*.
   The Rule Editor dialog box displays.
4. Type a *Name* and optionally a *Description* for the rule.
5. Verify the *Enabled* check box is selected.
6. For *Action on Fail*, select *Send to Fail*.
7. Select the *Column Validation* radio button.
   a) Select the *Column* CUSTOMERS.PHONE from the drop-down list.
   b) For *Condition*, from the drop-down list select *Match pattern*.
   c) For the value, type the expression `'99.99.99.99'`.
8. Click *OK*.
   The rule appears in the Rules list.

After running the job, the incorrectly formatted rows appear in the Fail output. You can now review the failed data, make corrections as necessary upstream, and reload the data.

Related Information

Analyzing the column profile [page 450]

2.15.4 Using Auditing

Auditing provides a way to ensure that a data flow loads correct data into the warehouse. Use auditing to perform the following tasks:

- Define audit points to collect run time statistics about the data that flows out of objects. Auditing stores these statistics in the repository.
- Define rules with these audit statistics to ensure that the data at the following points in a data flow is what you expect:
  - Extracted from sources

- Processed by transforms
- Loaded into targets

- Generate a run time notification that includes the audit rule that failed and the values of the audit statistics at the time of failure.
- Display the audit statistics after the job execution to help identify the object in the data flow that might have produced incorrect data.

Note

If you add an audit point prior to an operation that is usually pushed down to the database server, performance might degrade because push-down operations cannot occur after an audit point.

2.15.4.1 Auditing objects in a data flow

You can collect audit statistics on the data that flows out of any object, such as a source, transform, or target. If a transform has multiple distinct or different outputs (such as Validation or Case), you can audit each output independently.

To use auditing, you define the following objects in the *Audit* window:

<table>
<tr><th>Object name</th><th>Description</th></tr>
<tr><td>Audit point</td><td>The object in a data flow where you collect audit statistics. You can audit a source, a transform, or a target. You identify the object to audit when you define an audit function on it.</td></tr>
<tr><td>Audit function</td><td>The audit statistic that the software collects for a table, output schema, or column. The following table shows the audit functions that you can define.
<table>
<tr><th>Data object</th><th>Audit function</th><th>Description</th></tr>
<tr><td>Table or output schema</td><td>Count</td><td>This function collects two statistics:
• Good count for rows that were successfully processed.
• Error count for rows that generated some type of error if you enabled error handling.</td></tr>
<tr><td>Column</td><td>Sum</td><td>Sum of the numeric values in the column. Applicable data types include decimal, double, integer, and real. This function only includes the Good rows.</td></tr>
<tr><td>Column</td><td>Average</td><td>Average of the numeric values in the column. Applicable data types include decimal, double, integer, and real. This function only includes the Good rows.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Object name</th><th>Description</th></tr>
<tr><td></td><td><table><tr><th>Data object</th><th>Audit function</th><th>Description</th></tr><tr><td>Column</td><td>Checksum</td><td>Checksum of the values in the column.</td></tr></table></td></tr>
<tr><td>Audit label</td><td>The unique name in the data flow that the software generates for the audit statistics collected for each audit function that you define. You use these labels to define audit rules for the data flow.</td></tr>
<tr><td>Audit rule</td><td>A Boolean expression in which you use audit labels to verify the job. If you define multiple rules in a data flow, all rules must succeed or the audit fails.</td></tr>
<tr><td>Actions on audit failure</td><td>One or more of three ways to generate notification of an audit rule (or rules) failure: email, custom script, raise exception.</td></tr>
</table>

2.15.4.1.1 Audit function

This section describes the data types for the audit functions and the error count statistics.

Data types

The following table shows the default data type for each audit function and the permissible data types. You can change the data type in the *Properties* window for each audit function in the Designer.

Audit functions	Default data type	Allowed data types
Count	Integer	Integer
Sum	Type of audited column	Integer, Decimal, Double, Real
Average	Type of audited column	Integer, Decimal, Double, Real
Checksum	Varchar(128)	Varchar(128)

Error count statistic

When you enable a Count audit function, the software collects two types of statistics:

- *Good* row count for rows processed without any error.
- *Error* row count for rows that the job could not process but ignores those rows to continue processing. One way that error rows can result is when you specify the *Use overflow file* option in the Source Editor or Target Editor.

2.15.4.1.2 Audit label

The software generates a unique name for each audit function that you define on an audit point. You can edit the label names. You might want to edit a label name to create a shorter meaningful name or to remove dashes, which are allowed in column names but not in label names.

Generating label names

If the audit point is on a table or output schema, the software generates the following two labels for the audit function Count:

```
$Count<_objectname>
$CountError<_objectname>
```

If the audit point is on a column, the software generates an audit label with the following format:

```
$ <auditfunction_objectname>
```

If the audit point is in an embedded data flow, the labels have the following formats:

```
$Count<_objectname_embeddedDFname>
$CountError<_objectname_embeddedDFname>
$<auditfunction_objectname_embeddedDFname>
```

Editing label names

You can edit the audit label name when you create the audit function and before you create an audit rule that uses the label.

If you edit the label name after you use it in an audit rule, the audit rule does not automatically use the new name. You must redefine the rule with the new name.

2.15.4.1.3 Audit rule

An audit rule is a Boolean expression which consists of a Left-Hand-Side (LHS), a Boolean operator, and a Right-Hand-Side (RHS).

- The LHS can be a single audit label, multiple audit labels that form an expression with one or more mathematical operators, or a function with audit labels as parameters.
- The RHS can be a single audit label, multiple audit labels that form an expression with one or more mathematical operators, a function with audit labels as parameters, or a constant.

The following Boolean expressions are examples of audit rules:

```
$Count_CUSTOMER = $Count_CUSTDW
$Sum_ORDER_US + $Sum_ORDER_EUROPE = $Sum_ORDER_DW
round($Avg_ORDER_TOTAL) >= 10000
```

2.15.4.1.4 Audit notification

You can choose any combination of the following actions for notification of an audit failure. If you choose all three actions, the software executes them in this order:

- Email to list — the software sends a notification of which audit rule failed to the email addresses that you list in this option. Use a comma to separate the list of email addresses.
  You can specify a variable for the email list.
  This option uses the smtp_to function to send email. Therefore, you must define the server and sender for the Simple Mail Transfer Protocol (SMTP) in the Server Manager.
- Script — the software executes the custom script that you create in this option.
- Raise exception — The job fails if an audit rule fails, and the error log shows which audit rule failed. The job stops at the first audit rule that fails. This action is the default.
  You can use this audit exception in a try/catch block. You can continue the job execution in a try/catch block.
  If you clear this action and an audit rule fails, the job completes successfully and the audit does not write messages to the job log. You can view which rule failed in the Auditing Details report in the Metadata Reporting tool.

Related Information

Viewing audit results [page 463]

2.15.4.2 Accessing the Audit window

Access the *Audit* window from one of the following places in the Designer:

- From the Data Flows tab of the object library, right-click on a data flow name and select the *Auditing* option.
- In the workspace, right-click on a data flow icon and select the *Auditing* option.
- When a data flow is open in the workspace, click the *Audit* icon in the toolbar.

When you first access the *Audit* window, the Label tab displays the sources and targets in the data flow. If your data flow contains multiple consecutive query transforms, the *Audit* window shows the first query.

Click the icons on the upper left corner of the Label tab to change the display.

Icon	Tool tip	Description
⊟	Collapse All	Collapses the expansion of the source, transform, and target objects.

Icon	Tool tip	Description
	Show All Objects	Displays all the objects within the data flow.
	Show Source, Target and first-level Query	Default display which shows the source, target, and first-level query objects in the data flow. If the data flow contains multiple consecutive query transforms, only the first-level query displays.
Σ	Show Labelled Objects	Displays the objects that have audit labels defined.

2.15.4.3 Defining audit points, rules, and action on failure

1. Access the *Audit* window.
2. Define audit points.

   On the Label tab, right-click on an object that you want to audit and choose an audit function or Properties.

   When you define an audit point, the software generates the following:

   ○ An audit icon on the object in the data flow in the workspace
   ○ An audit label that you use to define audit rules.

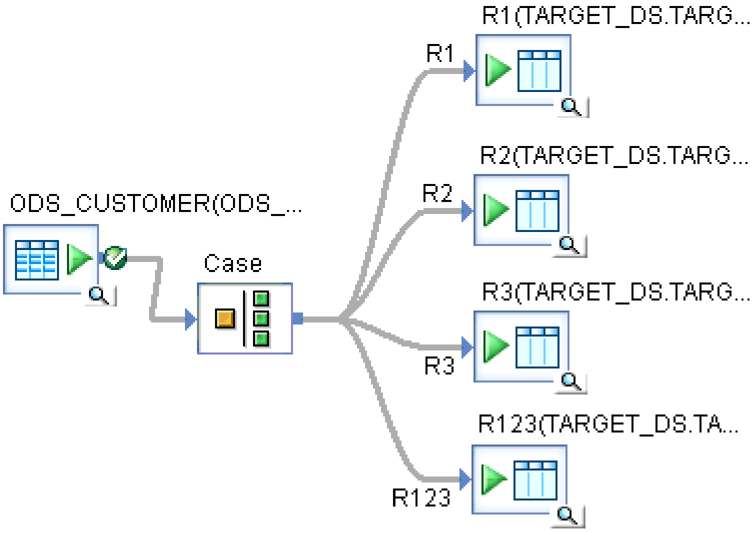
   In addition to choosing an audit function, the Properties window allows you to edit the audit label and change the data type of the audit function.

   For example, the data flow Case_DF has the following objects and you want to verify that all of the source rows are processed by the Case transform.

   ○ Source table ODS_CUSTOMER
   ○ Four target tables:
     R1 contains rows where ODS_CUSTOMER.REGION_ID = 1
     R2 contains rows where ODS_CUSTOMER.REGION_ID = 2
     R3 contains rows where ODS_CUSTOMER.REGION_ID = 3
     R123 contains rows where ODS_CUSTOMER.REGION_ID IN (1, 2 or 3)

   a) Right-click on source table ODS_CUSTOMER and choose *Count*.

      The software creates the audit labels $Count_ODS_CUSTOMER and $CountError_ODS_CUSTOMER, and an audit icon appears on the source object in the workspace.



b) Similarly, right-click on each of the target tables and choose *Count*. The Audit window shows the following audit labels.

Target table	Audit function	Audit label
ODS_CUSTOMER	Count	$Count_ODS_CUSTOMER
R1	Count	$Count_ R1
R2	Count	$Count_ R2
R3	Count	$Count_ R3
R123	Count	$Count_ R123

c) If you want to remove an audit label, right-click on the label, and the audit function that you previously defined displays with a check mark in front of it. Click the function to remove the check mark and delete the associated audit label.

When you right-click on the label, you can also select Properties, and select the value (No Audit) in the *Audit function* drop-down list.

3. Define audit rules. On the Rule tab in the *Audit* window, click *Add* which activates the expression editor of the Auditing Rules section.

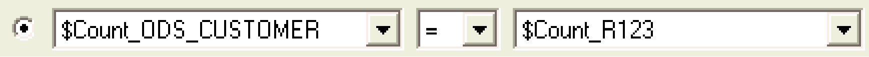
If you want to compare audit statistics for one object against one other object, use the expression editor, which consists of three text boxes with drop-down lists:

a) Select the label of the first audit point in the first drop-down list.

b) Choose a Boolean operator from the second drop-down list. The options in the editor provide common Boolean operators. If you require a Boolean operator that is not in this list, use the Custom expression box with its function and smart editors to type in the operator.

c) Select the label for the second audit point from the third drop-down list. If you want to compare the first audit value to a constant instead of a second audit value, use the Customer expression box.

For example, to verify that the count of rows from the source table is equal to the rows in the target table, select audit labels and the Boolean operation in the expression editor as follows:


● $Count_ODS_CUSTOMER = $Count_R123

If you want to compare audit statistics for one or more objects against statistics for multiple other objects or a constant, select the Custom expression box.

a) Click the ellipsis button to open the full-size smart editor window.
b) Click the *Variables* tab on the left and expand the *Labels* node.
c) Drag the first audit label of the object to the editor pane.
d) Type a Boolean operator
e) Drag the audit labels of the other objects to which you want to compare the audit statistics of the first object and place appropriate mathematical operators between them.
f) Click *OK* to close the smart editor.
g) The audit rule displays in the Custom editor. To update the rule in the top Auditing Rule box, click on the title "Auditing Rule" or on another option.
h) Click *Close* in the Audit window.

For example, to verify that the count of rows from the source table is equal to the sum of rows in the first three target tables, drag the audit labels, type in the Boolean operation and plus signs in the smart editor as follows:

```
Count_ODS_CUSTOMER = $Count_R1 + $Count_R2 + $Count_R3
```

4. Define the action to take if the audit fails.

   You can choose one or more of the following actions:

   - Raise exception: The job fails if an audit rule fails and the error log shows which audit rule failed. This action is the default.
     If you clear this option and an audit rule fails, the job completes successfully and the audit does not write messages to the job log. You can view which rule failed in the Auditing Details report in the Metadata Reporting tool.
   - Email to list: The software sends a notification of which audit rule failed to the email addresses that you list in this option. Use a comma to separate the list of email addresses.
     You can specify a variable for the email list.
   - Script: The software executes the script that you create in this option.

5. Execute the job.

   The *Execution Properties* window has the *Enable auditing* option checked by default. Clear this box if you do not want to collect audit statistics for this specific job execution.

6. Look at the audit results.

   You can view passed and failed audit rules in the metadata reports. If you turn on the audit trace on the Trace tab in the *Execution Properties* window, you can view all audit results on the Job Monitor Log.

Related Information

Auditing objects in a data flow [page 453]
Viewing audit results [page 463]

2.15.4.4 Guidelines to choose audit points

The following are guidelines to choose audit points:

- When you audit the output data of an object, the optimizer cannot push down operations after the audit point. Therefore, if the performance of a query that is pushed to the database server is more important than gathering audit statistics from the source, define the first audit point on the query or later in the data flow. For example, suppose your data flow has a source, query, and target objects, and the query has a WHERE clause that is pushed to the database server that significantly reduces the amount of data that returns to the software. Define the first audit point on the query, rather than on the source, to obtain audit statistics on the query results.
- If a pushdown_sql function is after an audit point, the software cannot execute it.
- You can only audit a bulkload that uses the Oracle API method. For the other bulk loading methods, the number of rows loaded is not available to the software.
- Auditing is disabled when you run a job with the debugger.
- You cannot audit NRDM schemas or real-time jobs.
- You cannot audit within an ABAP data flow, but you can audit the output of an ABAP data flow.
- If you use the CHECKSUM audit function in a job that normally executes in parallel, the software disables the DOP for the whole data flow. The order of rows is important for the result of CHECKSUM, and DOP processes the rows in a different order than in the source.

2.15.4.5 Auditing embedded data flows

You can define audit labels and audit rules in an embedded data flow. This section describes the following considerations when you audit embedded data flows:

- Enabling auditing in an embedded data flow
- Audit points not visible outside of the embedded data flow

2.15.4.5.1 Enabling auditing in an embedded data flow

If you want to collect audit statistics on an embedded data flow when you execute the parent data flow, you must enable the audit label of the embedded data flow.

2.15.4.5.1.1 To enable auditing in an embedded data flow

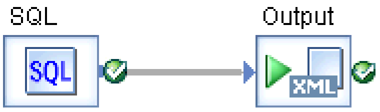
1. Open the parent data flow in the Designer workspace.
2. Click on the Audit icon in the toolbar to open the Audit window
3. On the Label tab, expand the objects to display any audit functions defined within the embedded data flow. If a data flow is embedded at the beginning or at the end of the parent data flow, an audit function might exist on the output port or on the input port.

4. Right-click the Audit function name and choose *Enable*. You can also choose *Properties* to change the label name and enable the label.
5. You can also define audit rules with the enabled label.

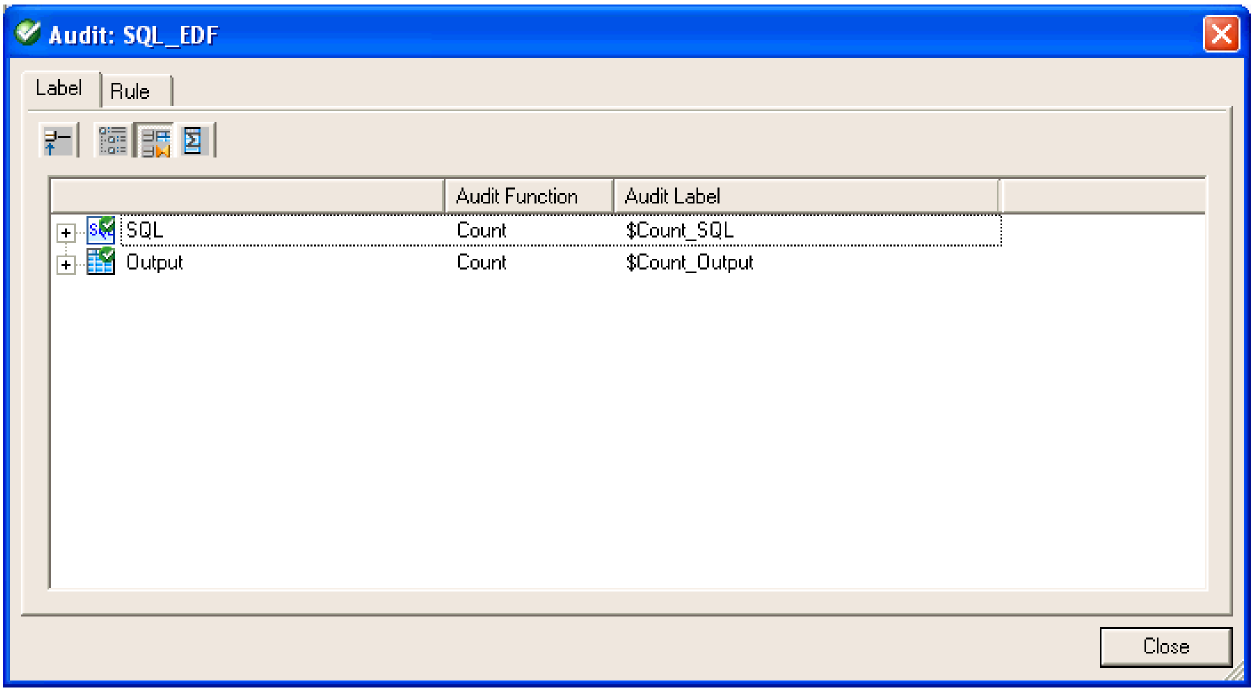
2.15.4.5.2 Audit points not visible outside of the embedded data flow

When you embed a data flow at the beginning of another data flow, data passes from the embedded data flow to the parent data flow through a single source. When you embed a data flow at the end of another data flow, data passes into the embedded data flow from the parent through a single target. In either case, some of the objects are not visible in the parent data flow.

Because some of the objects are not visible in the parent data flow, the audit points on these objects are also not visible in the parent data flow. For example, the following embedded data flow has an audit function defined on the source SQL transform and an audit function defined on the target table.

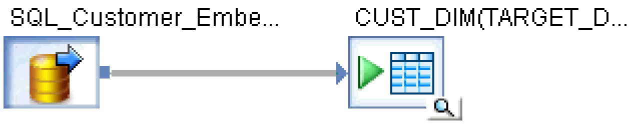


The following Audit window shows these two audit points.

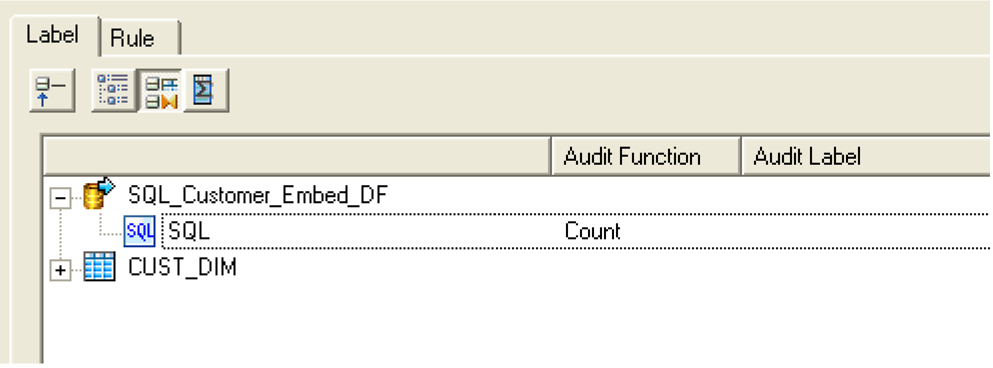


	Audit Function	Audit Label
SQL	Count	$Count_SQL
Output	Count	$Count_Output

When you embed this data flow, the target Output becomes a source for the parent data flow and the SQL transform is no longer visible.



An audit point still exists for the entire embedded data flow, but the label is no longer applicable. The following Audit window for the parent data flow shows the audit function defined in the embedded data flow, but does not show an Audit Label.



	Audit Function	Audit Label
SQL_Customer_Embed_DF		
SQL	Count	
CUST_DIM		

If you want to audit the embedded data flow, right-click on the audit function in the Audit window and select *Enable*.

2.15.4.6 Resolving invalid audit labels

An audit label can become invalid in the following situations:

- If you delete the audit label in an embedded data flow that the parent data flow has enabled.
- If you delete or rename an object that had an audit point defined on it

2.15.4.6.1 To resolve invalid audit labels

1. Open the Audit window.
2. Expand the Invalid Labels node to display the individual labels.
3. Note any labels that you would like to define on any new objects in the data flow.
4. After you define a corresponding audit label on a new object, right-click on the invalid label and choose *Delete*.
5. If you want to delete all of the invalid labels at once, right click on the Invalid Labels node and click on *Delete All*.

2.15.4.7 Viewing audit results

You can see the audit status in one of the following places:

- Job Monitor Log
- If the audit rule fails, the places that display audit information depends on the *Action on failure* option that you chose:

Action on failure	Places where you can view audit information
Raise exception	Job Error Log, Metadata Reports
Email to list	Email message, Metadata Reports
Script	Wherever the custom script sends the audit messages, Metadata Reports

Related Information

Job Monitor Log [page 463]
Job Error Log [page 464]
Metadata Reports [page 464]

2.15.4.7.1 Job Monitor Log

If you set *Audit Trace* to `Yes` on the Trace tab in the Execution Properties window, audit messages appear in the Job Monitor Log. You can see messages for audit rules that passed and failed.

The following sample audit success messages appear in the Job Monitor Log when *Audit Trace* is set to `Yes`:

```
Audit Label $Count_R2 = 4. Data flow <Case_DF>.
Audit Label $CountError_R2 = 0. Data flow <Case_DF>.
Audit Label $Count_R3 = 3. Data flow <Case_DF>.
Audit Label $CountError_R3 = 0. Data flow <Case_DF>.
Audit Label $Count_R123 = 12. Data flow <Case_DF>.
Audit Label $CountError_R123 = 0. Data flow <Case_DF>.
Audit Label $Count_R1 = 5. Data flow <Case_DF>.
Audit Label $CountError_R1 = 0. Data flow <Case_DF>.
Audit Label $Count_ODS_CUSTOMER = 12. Data flow <Case_DF>.
Audit Label $CountError_ODS_CUSTOMER = 0. Data flow <Case_DF>.
Audit Rule passed ($Count_ODS_CUSTOMER = (($CountR1 + $CountR2 + $Count_R3)):
LHS=12, RHS=12. Data flow <Case_DF>.
Audit Rule passed ($Count_ODS_CUSTOMER = $CountR123): LHS=12, RHS=12. Data flow
<Case_DF>.
```

2.15.4.7.2 Job Error Log

When you choose the *Raise exception* option and an audit rule fails, the Job Error Log shows the rule that failed. The following sample message appears in the Job Error Log:

```
Audit rule failed <($Count_ODS_CUSTOMER = $CountR1)> for <Data flow Case_DF>.
```

2.15.4.7.3 Metadata Reports

You can look at the Audit Status column in the Data Flow Execution Statistics reports of the Metadata Report tool. This Audit Status column has the following values:

- Not Audited
- Passed—All audit rules succeeded. This value is a link to the Auditing Details report which shows the audit rules and values of the audit labels.
- Information Collected—This status occurs when you define audit labels to collect statistics but do not define audit rules. This value is a link to the Auditing Details report which shows the values of the audit labels.
- Failed—Audit rule failed. This value is a link to the Auditing Details report which shows the rule that failed and values of the audit labels.

Related Information

Management Console Guide: Operational Dashboard Reports [page 1975]

2.16 Data Quality

2.16.1 Overview of data quality

Data quality is a term that refers to the set of transforms that work together to improve the quality of your data by cleansing, enhancing, matching and consolidating data elements.

Data quality is primarily accomplished in the software using four transforms:

- Address Cleanse: Parses, standardizes, corrects, and enhances address data.
- Data Cleanse: Parses, standardizes, corrects, and enhances customer and operational data.
- Geocoding: Uses geographic coordinates, addresses, and point-of-interest (POI) data to append address, latitude and longitude, census, and other information to your records.
- Match: Identifies duplicate records at multiple levels within a single pass for individuals, households, or corporations within multiple tables or databases and consolidates them into a single source.

Related Information

Address Cleanse [page 585]
About cleansing data [page 465]
Geocoding [page 489]
Match [page 510]

2.16.2 Data Cleanse

2.16.2.1 About cleansing data

Data cleansing is the process of parsing and standardizing data.

The parsing rules and other information that define how to parse and standardize are stored in a cleansing package. The Cleansing Package Builder in SAP Information Steward provides a graphical user interface to create and refine cleansing packages. You can create a cleansing package from scratch based on sample data or adapt an existing cleansing package or SAP-supplied cleansing package to meet your specific data cleansing requirements and standards.

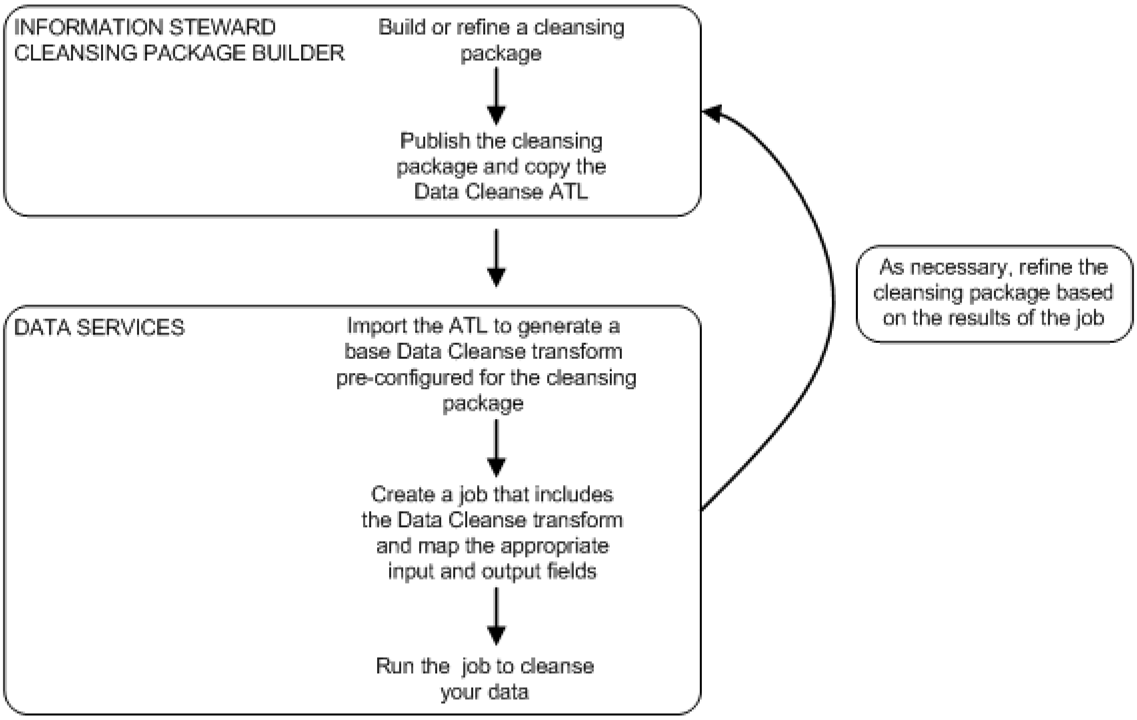
> Note
>
> The Data Cleansing Advisor feature uses the SAP-supplied Person_Firm cleansing package when processing data. Do not delete or rename the SAP-supplied cleansing package. If you modify the SAP-supplied cleansing package, it may alter the results in Data Cleansing Advisor.

> Note
>
> After creating one or more custom content types in Data Insight, you will see a cleansing package named CTID_CUSTOM_TYPE in Cleansing Package Builder. Do not open or modify CTID_CUSTOM_TYPE in Cleansing Package Builder because it will impact the custom content types.

A cleansing package is created and published within Cleansing Package Builder and then referenced by the Data Cleanse transform within Data Services for testing and production deployment.

Within a Data Services work flow, the Data Cleanse transform identifies and isolates specific parts of mixed data, and then parses and formats the data based on the referenced cleansing package as well as options set directly in the transform.

The following diagram shows how Data Services and Information Steward work together to allow you to develop a cleansing package specific to your data requirements and then apply it when you cleanse your data.


INFORMATION STEWARD CLEANSING PACKAGE BUILDER
Build or refine a cleansing package
Publish the cleansing package and copy the Data Cleanse ATL
DATA SERVICES
Import the ATL to generate a base Data Cleanse transform pre-configured for the cleansing package
Create a job that includes the Data Cleanse transform and map the appropriate input and output fields
Run the job to cleanse your data
As necessary, refine the cleansing package based on the results of the job

2.16.2.2 Cleansing package lifecycle: develop, deploy and maintain

The process of developing, deploying, and maintaining a cleansing package is the result of action and communication between the Data Services administrator, Data Services tester, and Cleansing Package Builder data steward. The exact roles, responsibilities, and titles vary by organization, but often include the following:

Role	Responsibility
Cleansing Package Builder data steward	Uses Cleansing Package Builder and has domain knowledge to develop and refine a cleansing package for a specific data domain.
Data Services tester	In a Data Services test environment, uses the Data Services transform to cleanse data and verify the results. Works with the Cleansing Package Builder data steward to refine a cleansing package.
Data Services administrator	In a Data Services production environment, uses the Data Cleanse transform to cleanse data based on the rules and standards defined in the selected cleansing package.

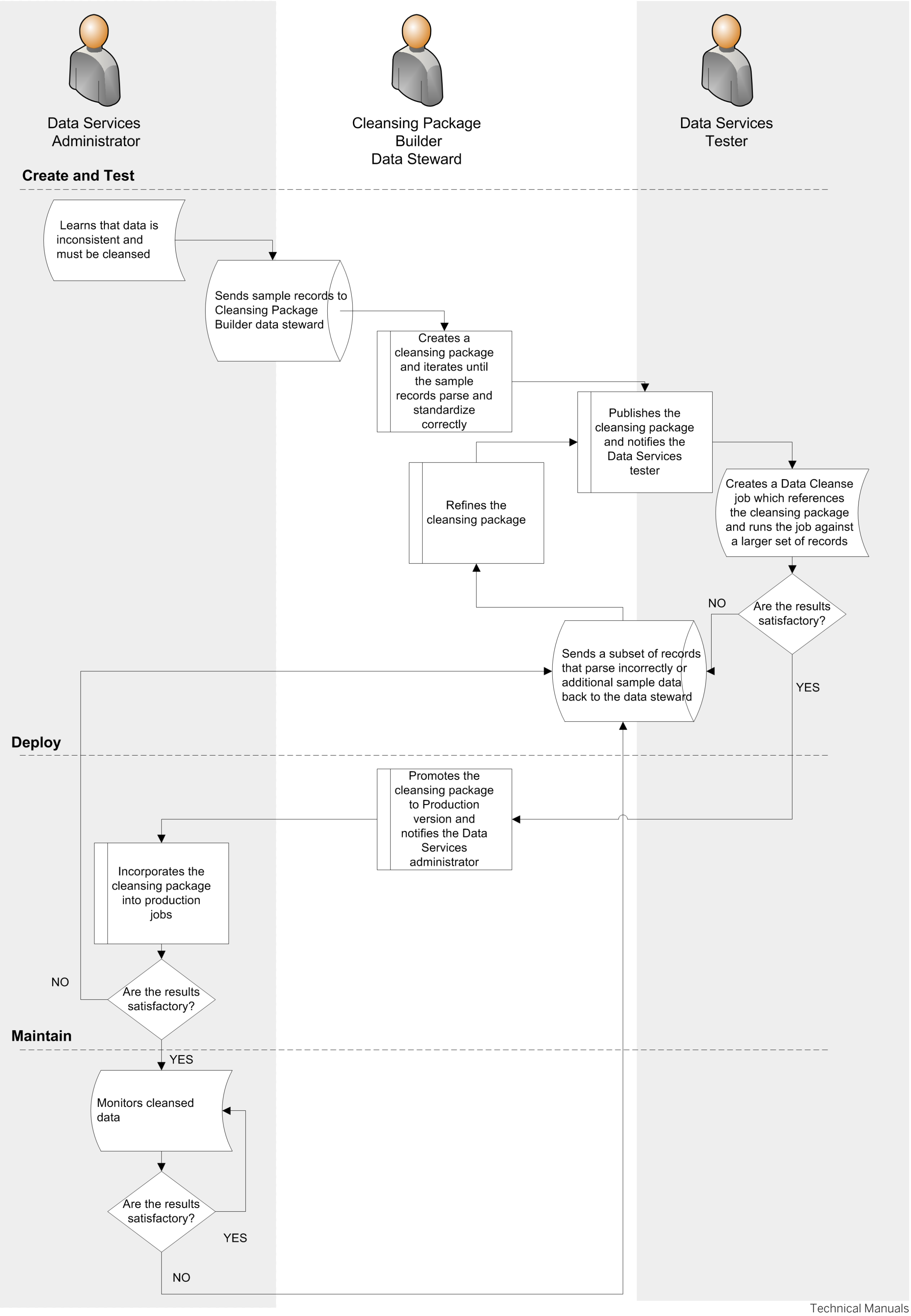
There are typically three iterative phases in a cleansing package workflow: develop (create and test), deploy, and maintain.

In the create and test phase, the data steward creates a custom cleansing package based on sample data provided by the Data Services administrator and then works with the Data Services tester to refine the cleansing package. When everyone is satisfied with the results, the cleansing package is deployed to production.

In the deployment phase the Data Services administrator, tester, and data steward work together to further refine the cleansing package so that production data is cleansed within the established acceptable range.

Finally, the cleansing package is moved to the maintenance phase and updated only when the results of regularly scheduled jobs fall out of range or when new data is introduced.

A typical workflow is shown in the diagram below:


Data Services Administrator
Cleansing Package Builder Data Steward
Data Services Tester
Create and Test
Learns that data is inconsistent and must be cleansed
Sends sample records to Cleansing Package Builder data steward
Creates a cleansing package and iterates until the sample records parse and standardize correctly
Publishes the cleansing package and notifies the Data Services tester
Refines the cleansing package
Creates a Data Cleanse job which references the cleansing package and runs the job against a larger set of records
Are the results satisfactory?
NO
YES
Sends a subset of records that parse incorrectly or additional sample data back to the data steward
Deploy
Promotes the cleansing package to Production version and notifies the Data Services administrator
Incorporates the cleansing package into production jobs
NO
Are the results satisfactory?
Maintain
YES
Monitors cleansed data
Are the results satisfactory?
YES
NO

For more information about the process of moving a cleansing package from development to production to maintenance, see the *Information Steward Administrator Guide.*

For more information about how to use the promotion management tool in the Central Management Console (CMC) to move cleansing packages (referred to as promoting "Objects" or "InfoObjects"), see the *Business Intelligence platform Administrator Guide.*

2.16.2.3 Configuring the Data Cleanse transform

Prerequisites for configuring the Data Cleanse transform include:

- Access to the necessary cleansing package.
- Access to the ATL file transferred from Cleansing Package Builder.
- Input field and attribute (output field) mapping information for user-defined pattern matching rules defined in the *Reference Data* tab of Cleansing Package Builder.

To configure the Data Cleanse transform:

1. Import the ATL file transferred from Cleansing Package Builder.

   Importing the ATL file brings the required information and automatically sets the following options:

   - Cleansing Package
   - Filter Output Fields
   - Input Word Breaker
   - Parser Configuration

   > Note
   >
   > You can install and use SAP-supplied person and firm cleansing package without modifications directly in Data Services. To do so, skip step 1 and manually set any required options in the Data Cleanse transform.

2. In the input schema, select the input fields that you want to map and drag them to the appropriate fields in the *Input* tab.

   - Name and firm data can be mapped either to discrete fields or multiline fields.
   - Custom data must be mapped to multiline fields.
   - Phone, date, email, Social Security number, and user-defined pattern data can be mapped either to discrete fields or multiline fields. The corresponding parser must be enabled.

3. In the *Options* tab, select the appropriate option values to determine how Data Cleanse will process your data.

   If you change an option value from its default value, a green triangle appears next to the option name to indicate that the value has been changed.

   The ATL file that you imported in step 1 sets certain options based on information in the cleansing package.

4. In the *Output* tab, select the fields that you want to output from the transform. In Cleansing Package Builder, output fields are referred to as attributes.

   Ensure that you map any attributes (output fields) defined in user-defined patterns in Cleansing Package Builder reference data.

Related Information

Transform configurations [page 308]
Data Quality transform editors [page 314]
To add a Data Quality transform to a data flow [page 312]

2.16.2.4 Ranking and prioritizing parsing engines

When dealing with multiline input, you can configure the Data Cleanse transform to use only specific parsers and to specify the order the parsers are run. Carefully selecting which parsers to use and in what order can be beneficial. Turning off parsers that you do not need significantly improves parsing speed and reduces the chances that your data will be parsed incorrectly.

You can change the parser order for a specific multiline input by modifying the corresponding parser sequence option in the Parser_Configuration options group of the Data Cleanse transform. For example, to change the order of parsers for the Multiline1 input field, modify the Parser_Sequence_Multiline1 option.

To change the selected parsers or the parser order: select a parser sequence, click OK at the message and then use the Ordered Options window to make your changes.

Related Information

Ordered options editor [page 317]

2.16.2.5 About parsing data

The Data Cleanse transform can identify and isolate a wide variety of data. Within the Data Cleanse transform, you map the input fields in your data to the appropriate input fields in the transform. Custom data containing operational or product data is always mapped to multiline fields. Person and firm data, phone, date, email, and Social Security number data can be mapped to either discrete input fields or multiline input fields.

The example below shows how Data Cleanse parses product data from a multiline input field and displays it in discrete output fields. The data also can be displayed in composite fields, such as *Standard Description*, which can be customized in Cleansing Package Builder to meet your needs.

Input data	Parsed data	
Glove ultra grip profit 2.3 large black synthetic leather elastic with Velcro Mechanix Wear	Product Category	Glove
	Size	Large
	Material	Synthetic Leather
	Trademark	Pro-Fit 2.3 Series

Input data	Parsed data	
	Cuff Style	Elastic Velcro
	Palm Type	Ultra-Grip
	Color	Black
	Vendor	Mechanix Wear
	Standard Description	Glove - Synthetic Leather, Black, size: Large, Cuff Style: Elastic Velcro, Ultra-Grip, Mechanix Wear

The examples below show how Data Cleanse parses name and firm data and displays it in discrete output fields. The data also can be displayed in composite fields which can be customized in Cleansing Package Builder to meet your needs.

Input data	Parsed data	

`Mr. Dan R. Smith, Jr., CPA`
`Account Mgr.`
`Jones Inc.`
	Prename	Mr.

	Given Name 1	Dan
	Given Name 2	R.
	Family Name 1	Smith
	Maturity Postname	Jr.
	Honorary Postname	CPA
	Title	Account Mgr.
	Firm	Jones, Inc.

Input data	Parsed data	

`James Witt`
`421-55-2424`
`jwitt@rdrindustries.com`
`507-555-3423`
`Aug 20, 2003`
	Given Name 1	James

	Family Name 1	Witt
	Social Security	421-55-2424
	E-mail address	jwitt@rdrindustries.com
	Phone	507.555.3423
	Date	August 20, 2003

The Data Cleanse transform parses up to six names per record, two per input field. For all six names found, it parses components such as prename, given names, family name, and postname. Then it sends the data to individual fields. The Data Cleanse transform also parses up to six job titles per record.

The Data Cleanse transform parses up to six firm names per record, one per input field.

2.16.2.5.1 About parsing phone numbers

Data Cleanse can parse both North American Numbering Plan (NANP) and international phone numbers. When Data Cleanse parses a phone number, it outputs the individual components of the number into the appropriate fields.

Phone numbering systems differ around the world. Data Cleanse recognizes phone numbers by their pattern and (for non-NANP numbers) by their country code, too.

Data Cleanse searches for North American phone numbers by commonly used patterns such as (234) 567-8901, 234-567-8901, and 2345678901. Data Cleanse gives you the option for some reformatting on output (such as your choice of delimiters).

Data Cleanse searches for non-North American numbers by pattern. The patterns used are specified in Cleansing Package Builder in the *Reference Data* tab. The country code must appear at the beginning of the number. Data Cleanse does not offer any options for reformatting international phone numbers. Also, Data Cleanse does not cross-compare to the address to see whether the country and city codes in the phone number match the address.

Related Information

About one-to-one mapping [page 477]

2.16.2.5.2 About parsing dates

Data Cleanse recognizes dates in a variety of formats and breaks those dates into components.

Data Cleanse can parse up to six dates from your defined record. That is, Data Cleanse identifies up to six dates in the input, breaks those dates into components, and makes dates available as output in either the original format or a user-selected standard format.

Related Information

About one-to-one mapping [page 477]

2.16.2.5.3 About parsing Social Security numbers

Data Cleanse parses U.S. Social Security numbers (SSNs) that are either by themselves or on an input line surrounded by other text.

Fields used

Data Cleanse outputs the individual components of a parsed Social Security number—that is, the entire SSN, the area, the group, and the serial.

How Data Cleanse parses Social Security numbers

Data Cleanse parses Social Security numbers in the following steps:

1. Identifies a potential SSN by looking for the following patterns:

Pattern	Digits per grouping	Delimited by
nnnnnnnnn	9 consecutive digits	not applicable
nnn nn nnnn	3, 2, and 4 (for area, group, and serial)	spaces
nnn-nn-nnnn	3, 2, and 4 (for area, group, and serial)	all supported delimiters

2. Performs a validity check on the first five digits only. The possible outcomes of this validity check are:

Outcome	Description
Pass	Data Cleanse successfully parses the data—and the Social Security number is output to a SSN output field.
Fail	Data Cleanse does not parse the data because it is not a valid Social Security number as defined by the U.S. government. The data is output as Extra, unparsed data.

Check validity

When performing a validity check, Data Cleanse does not verify that a particular 9-digit Social Security number has been issued, or that it is the correct number for any named person. Instead, it validates only the first 5 digits (area and group). Data Cleanse does not validate the last 4 digits (serial)—except to confirm that they are digits.

SSA data

Data Cleanse validates the first five digits based on a table from the Social Security Administration (*http://www.socialsecurity.gov/employer/ssns/HGJune2411_final.txt*). The rules and data that guide this check are available at *http://www.ssa.gov/history/ssn/geocard.html* . The Social Security number information that Data Cleanse references is included in the cleansing package.

> Note
>
> The Social Security administration no longer updates the table. The last time it was updated was in July, 2011. Therefore, Social Security validation performed in Data Cleans is based on data through July, 2011. For more

information about the Social Security Administration's Social Security number assignment process, see *http://www.ssa.gov/employer/randomizationfaqs.html* .

Outputs valid SSNs

Data Cleanse outputs only Social Security numbers that pass its validation. If an apparent SSN fails validation, Data Cleanse does not pass on the number as a parsed, but invalid, Social Security number.

Related Information

Reference Guide: Transforms, Data Cleanse output fields [page 1164]

2.16.2.5.4 About parsing email addresses

When Data Cleanse parses input data that it determines is an email address, it places the components of that data into specific fields for output. Below is an example of a simple email address:

```
joex@sap.com
```

By identifying the various data components (user name, host, and so on) by their relationships to each other, Data Cleanse can assign the data to specific attributes (output fields).

Output fields Data Cleanse uses

Data Cleanse outputs the individual components of a parsed email address—that is, the email user name, complete domain name, top domain, second domain, third domain, fourth domain, fifth domain, and host name.

What Data Cleanse does

Data Cleanse can take the following actions:

- Parse an email address located either in a discrete field or combined with other data in a multiline field.
- Break down the domain name into sub-elements.
- Verify that an email address is properly formatted.
- Flag that the address includes an internet service provider (ISP) or email domain name listed in the email type of Reference Data in Data Cleanse. This flag is shown in the Email_is_ISP output field.

What Data Cleanse does not verify

Several aspects of an email address are not verified by Data Cleanse. Data Cleanse does not verify:

- whether the domain name (the portion to the right of the @ sign) is registered.
- whether an email server is active at that address.
- whether the user name (the portion to the left of the @ sign) is registered on that email server (if any).
- whether the personal name in the record can be reached at this email address.

Email components

The output field where Data Cleanse places the data depends on the position of the data in the record. Data Cleanse follows the Domain Name System (DNS) in determining the correct output field.

For example, if `expat@london.home.office.city.co.uk` were input data, Data Cleanse would output the elements in the following fields:

Output field	Output value
Email	expat@london.home.office.city.co.uk
Email_User	expat
Email_Domain_All	london.home.office.city.co.uk
Email_Domain_Top	uk
Email_Domain_Second	co
Email_Domain_Third	city
Email_Domain_Fourth	office
Email_Domain_Fifth	home
Email_Domain_Host	london

Related Information

About one-to-one mapping [page 477]

2.16.2.5.5 About parsing user-defined patterns

Data Cleanse can parse patterns found in a wide variety of data such as:

- account numbers
- part numbers

- purchase orders
- invoice numbers
- VINs (vehicle identification numbers)
- driver license numbers

In other words, Data Cleanse can parse any alphanumeric sequence for which you can define a pattern.

The user-defined pattern matching (UDPM) parser looks for the pattern across each entire field.

Patterns are defined using regular expressions in the *Reference Data* tab of Cleansing Package Builder. Check with the cleansing package owner to determine any required mappings for input fields and output fields (attributes).

2.16.2.5.6 About parsing street addresses

Data Cleanse does not identify and parse individual address components. To parse data that contains address information, process it using a Global Address Cleanse or U.S. Regulatory Address Cleanse transform prior to Data Cleanse. If address data is processed by the Data Cleanse transform, it is usually output to the *Extra* fields.

Related Information

How address cleanse works [page 586]

2.16.2.5.7 About parsing firm names

Data Cleanse can parse firm data.

Data Cleanse accepts these firm names alone in a field or together with other data.

An exception to how Data Cleanse recombines contiguous word pieces is made for words that end with an *S*, such as Applebee's or Macy's. An input string of *Macy's* is broken into three individual tokens: MACY, ', S. Because the last token is an *S*, Data Cleanse first combines the tokens and looks up the term including the apostrophe (MACY'S). If the term is not found, Data Cleanse looks up the term without the apostrophe (MACYS). If that is not successful, Data Cleanse automatically keeps the tokens together (MACY'S) and adds the FIRM_MISCELLANEOUS classification to the term. Since words ending with *S* are automatically kept together, it is not necessary to add all possessive firm names to the dictionary.

2.16.2.5.8 About parsing name and title data

Data Cleanse can parse name and title data.

A person's name can consist of the following parts: prename, given names, family names, postname, and so on.

Data Cleanse can accept up to two names and titles as discrete components. Data Cleanse also accepts name and title data together with other data or alone in a field. The name line or multiline field may contain one or two names per field.

2.16.2.5.9 About one-to-one mapping

One-to-one mapping is an option in the Data Cleanse transform that controls how several parsers output the data.

The *One-to-one mapping* option is available for these parsers:

- Date
- Email
- Phone

When the option is set to *Yes*, the Data Cleanse transform outputs the data parsed from certain discrete input fields to their corresponding output fields. The output fields are "reserved" for parses from certain discrete input fields. This option more clearly shows the mapping of the input field that held the original data based on the parsed data in the output field. For example, if the input data in Phone1 through Phone5 were blank and the Phone6 field contained data, then on output, Phone1 through Phone5 fields continue to be blank and Phone6 contains the parsed data.

When *One-to-one mapping* is set to *Yes*, then all parsers that use this option are set to *Yes*. For example, you cannot set the *One-to-one mapping* option only for the phone parser.

When *One-to-one mapping* is set to *No*, the Data Cleanse transform parses and outputs the data in the order the data entered the parser. The data is not necessarily output to the same field that it was mapped to on output. The data is output to the first available field in the category.

Note

The examples in this section show Date fields. The same examples also apply to Phone and Email fields.

Example

Field	Input data	Output data when option is *No*	Output data when option is *Yes*
Date1	<blank>	1968/01/01	<blank>
Date2	1968/01/01	1968/02/02	1968/01/01
Date3	1968/02/02	1968/03/03	1968/02/02
Date4	<blank>	1968/04/04	<blank>
Date5	1968/03/03	<blank>	1968/03/03
Date6	1968/04/04	<blank>	1968/04/04

Multiline fields

The discrete Date, Email, and Phone fields are parsed before the Multiline fields, so that any unreserved fields can contain data from the Multiline fields.

Example

Field	Input data	Output data when option is *No*	Output data when option is *Yes*
Date1	<blank>	1968/01/01	<blank>
Date2	1968/01/01	1968/02/02	1968/01/01
Date3	<blank>	1968/03/03	<blank>
Date4	1968/02/02	1968/04/04	1968/02/02
Date5	<blank>	<blank>	1968/03/03 (not reserved, so Multiline input can be added here)
Date6	<blank>	<blank>	1968/04/04 (not reserved, so Multiline input can be added here)
Multi-line1	1968/03/03 1968/04/04	<blank>	<blank>

Extra fields

When the *One-to-one mapping* option is set to *Yes* and the input field contains multiple sets of data, only the first set of data is placed in the corresponding output field. All other sets of data are put in the Extra field.

Example

Field	Input data	Output data when option is *No*	Output data when option is *Yes*
Date1	1968/01/01 1968/02/02 1968/03/03 1968/04/04 1968/05/05 1968/06/06 1968/07/07 1968/08/08	1968/01/01	1968/01/01
Date2	<blank>	1968/02/02	<blank>
Date3	<blank>	1968/03/03	<blank>
Date4	<blank>	1968/04/04	<blank>
Date5	<blank>	1968/05/05	<blank>

Field	Input data	Output data when option is *No*	Output data when option is *Yes*
Date6	<blank>	1968/06/06	<blank>
Extra	<blank>	1968/07/07 1968/08/08	1968/02/02 1968/03/03 1968/04/04 1968/05/05 1968/06/06 1968/07/07 1968/08/08

Related Information

About parsing dates [page 472]
About parsing email addresses [page 474]
About parsing phone numbers [page 472]

2.16.2.6 About standardizing data

Standard forms for individual variations are defined within a cleansing package using Cleansing Package Builder. Additionally, the Data Cleanse transform can standardize data to make its format more consistent. Data characteristics that the transform can standardize include case, punctuation, and abbreviations.

2.16.2.7 About assigning gender descriptions and prenames

Each variation in a cleansing package has a gender associated with it. By default, the gender is *unassigned*. You can assign a gender to a variation in the Advanced mode of Cleansing Package Builder. Gender descriptions are: strong male, strong female, weak male, weak female, and ambiguous.

Variations in the SAP-supplied name and firm cleansing package have been assigned genders.

You can use the Data Cleanse transform to output the gender associated with a variation to the GENDER output field.

The Prename output field always includes prenames that are part of the name input data. Additionally, when the *Assign Prenames* option is set to *Yes*, Data Cleanse populates the PRENAME output field when a strong male or strong female gender is assigned to a variation.

Dual names

When dual names are parsed, Data Cleanse offers four additional gender descriptions: female multi-name, male multi-name, mixed multi-name, and ambiguous multi-name. These genders are generated within Data Cleanse based on the assigned genders of the two names. The table below shows how the multi-name genders are assigned:

Dual name	Gender of first name	Gender of second name	Assigned gender for dual name
Bob and Sue Jones	strong male	strong female	mixed multi-name
Bob and Tom Jones	strong male	strong male	male multi-name
Sue and Sara Jones	strong female	strong female	female multi-name
Bob and Pat Jones	strong male	ambiguous	ambiguous multi-name

Chinese and Japanese given names

When a given name was parsed as the result of the rules intelligently combining given name characters as opposed to including the given name as a variation in the cleansing package, Data Cleanse generates the gender by combining the gender of the individual characters that make up the given name, using the table below.

	Strong female	*Strong male*	*Weak female*	*Weak male*	*Ambiguous*
Strong female	strong female	ambiguous	strong female	ambiguous	strong female
Strong male	ambiguous	strong male	ambiguous	strong male	strong male
Weak female	strong female	ambiguous	weak female	ambiguous	weak female
Weak male	ambiguous	strong male	ambiguous	weak male	weak male
Ambiguous	strong female	strong male	weak female	weak male	ambiguous

2.16.2.8 Prepare records for matching

If you are planning a data flow that includes matching, it is recommended that you first use Data Cleanse to standardize the data to enhance the accuracy of your matches. The Data Cleanse transform should be upstream from the Match transform.

The Data Cleanse transform can generate match standards or alternates for many name and firm fields as well as all custom output fields. For example, Data Cleanse can tell you that Patrick and Patricia are potential matches for the name Pat. Match standards can help you overcome two types of matching problems: alternate spellings (Catherine and Katherine) and nicknames (Pat and Patrick).

This example shows how Data Cleanse can prepare records for matching.

Table 8: Data source 1

Input record	Cleansed record	
Intl Marketing, Inc. Pat Smith, Accounting Mgr.	Given Name 1	Pat
	Match Standards	Patrick, Patricia
	Given Name 2	
	Family Name 1	Smith

Input record	Cleansed record	
	Title	Accounting Mgr.
	Firm	Intl. Mktg, Inc.

Table 9: Data source 2

Input record	Cleansed record	
Smith, Patricia R. International Marketing, Incorp.	Given Name 1	Patricia
	Match Standards	
	Given Name 2	R
	Family Name 1	Smith
	Title	
	Firm	Intl. Mktg, Inc.

When a cleansing package does not include an alternate, the match standard output field for that term will be empty. In the case of a multi-word output such as a firm name, when none of the variations in the firm name have an alternate, then the match standard output will be empty. However, if at least one variation has an alternate associated with it, the match standard is generated using the variation alternate where available and the variations for words that do not have an alternate.

2.16.2.9 Region-specific data

2.16.2.9.1 About domains

A domain describes a specific type of data or content. Domains enhance the ability of Data Cleanse to appropriately cleanse data according to the cultural standards of a region. Within an SAP-supplied person and firm cleansing package each supported locale is a domain. The table below illustrates how name parsing may vary by culture:

Domain	Name	Parsed Output			
		Given_Name1	Given_Name2	Family_Name1	Family_Name2
Spanish	Juan C. Sánchez	Juan	C.	Sánchez	
Portuguese	João A. Lopes	João		A.	Lopes
French	Jean Christophe Rousseau	Jean Christophe		Rousseau	
German	Hans Joachim Müller	Hans	Joachim	Müller	
English (U.S. and Canada)	James Andrew Smith	James	Andrew	Smith	

Each variation is automatically assigned to the Global domain and may also be assigned to one or more other domains. A variation may have a different meaning in each domain. In other words, the properties associated with a given variation, such as standard form and classification, may be specific to a domain. For example, the variation `AG` has different meanings in German and in U.S. English. In German, `AG` is an abbreviation for “Aktiengesellschaft” (a firm type) and is cleansed as "AG", while in English `AG` is an abbreviation for Agriculture and is cleansed as "Ag." You can control how Data Cleanse cleanses your data by specifying which domain or domains you want Data Cleanse to apply and in what order.

> Note
>
> Multiple domains are supported only in person and firm cleansing packages version 4.1 or higher. Variations in custom cleansing packages as well as person and firm cleansing packages created prior to Information Steward 4.1 are assigned only to the Global domain.

Global domain

The Global domain is a special content domain which contains all variations and their associated properties. If a variation is not associated with domain-specific information the Global domain serves as the default domain.

When you add a new variation, it is initially added to the Global domain. As necessary, you can then add it to other domains in order to assign any domain-specific information. You only need to add a variation to a domain other than the Global domain when the variation has properties such as gender, classification, standard form, and so on, which differ between the Global and other domains.

If you delete a variation from the Global domain, the variation is also deleted from all other domains it is associated with.

Controlling domain-specific cleansing

The Data Services administrator or tester can set the *Content Domain Sequence* in the Data Cleanse transform to control how Data Cleanse parses your domain-specific data such as name, gender title, and so on. In the examples below consider how gender would be applied for the name Jean based the following information:

Name	Domain	Gender
Jean	Global	AMBIGUOUS
Jean	French	STRONG_MALE
Jean	English (United States and Canada)	WEAK_FEMALE

When you do not want to favor any domain-specific properties, select only *GLOBAL*. The name Jean will be assigned an ambiguous gender because neither the French nor the English domain-specific information is considered.

When you have data from a single-domain region, specify a domain followed by Global. For example, when you specify EN_US followed by GLOBAL (*EN_US|GLOBAL*), the name Jean will be assigned a weak female gender.

When you have data from a multi-domain region, select the preferred sequence of domains ending with Global. For example, Benelux (Belgium, Netherlands, Luxembourg) includes the Dutch, French, and German domains. Depending on your cleansing preference you can order the domains in the desired sequence. For example, if you favor the Dutch domain you would specify *NL|FR|DE|GLOBAL*. When a variation is encountered in your data that has different properties in the selected content domains, the cleansing process uses the Dutch properties first if they exist. If there are no Dutch-specific properties then Data Cleanse uses the French properties, if there are neither Dutch-specific nor French-specific properties then it uses the German properties. If none of the three domains have specific properties, then Data Cleanse uses the properties that are specified in the Global domain.

Another example of a multi-domain region is Switzerland. Switzerland includes German, French, and Italian domains. Depending on your cleansing preference you can order the domains in the desired sequence. For example, if you favor the German domain you may select *DE|FR|IT|GLOBAL*. When a variation is encountered in your data that has different properties in the selected content domains, the cleansing process uses the German properties first if they exist. If there are no German-specific properties then Data Cleanse uses the French properties, if there are neither German-specific nor French-specific properties then it uses the Italian properties. If none of the three have a specific meaning then Data Cleanse uses the meaning that exists in the Global domain.

Specifying the content domain sequence

You can set the content domain sequence in three different levels.

Level	Option
Default	Data Cleanse *Content Domain Sequence* transform option. This option is required. If there are errors or the intermediate-level and top-level options are not specified, this setting is used.
Intermediate	The data in Option_Country, Option_Language, and/or Option_Region input options. If there are errors or the top-level option is not specified, this setting is used. These input fields are populated and mapped from the Global Address Cleanse transform output fields ISO_Country_Code_2Char, Language, and Region1, respectively. Using these input fields is optional.
Top	The data in Option_Content_Domain_Sequence input option. The data in this option is assigned through a preceding Query transform or other preprocessing step. Using this input field is optional.

Example

During processing, Data Cleanse assigns the content domain sequence based on the top-level option, if the option is set and is valid.

Level	Option	Output	Notes
Top	Input field: Option_Content_Domain_Sequence=FR|Global	The content domain sequence is French, and all of the data that isn't identified as French defaults to the Global domain.	This option overrides the intermediate and default settings, unless this option is invalid (for example, misspelled) or not set.

Intermediate	Input fields:
	Option_Country=MX
	Option_Language=<not set>
		The content domain sequence is Spanish (ES_MX, local to Mexico), and all of the data that isn't identified as Spanish	This option overrides the default setting, unless this option contains invalid data, or was not set.

Level	Option	Output	Notes
	Option_Region=<not set>	defaults to the Global domain.	In most cases, the content domain sequence can be identified based on the Option_Country input field only. However, there are certain countries that can use Option_Language or Option_Region to help determine the content domain sequence.
Default	Data Cleanse transform option: Content Domain Sequence=EN_US|Global	The content domain sequence is English (local to United States and Canada), and all of the data that isn't identified as English defaults to the Global domain.	This option is required and is used when there are errors in the top-level and intermediate-level options.

When using the intermediate level, you will find that some countries can use additional input data to determine the content domain sequence.

Option_Country input field	Additional input fields	Assigned content domain sequence	Notes

CA (Canada)	Option_Region=QC
	Option_Region=BC*
		FR|Global
		EN_US|Global
			The content domain sequence is always assigned to FR|Global when Option_Region is set to Quebec. Acceptable data includes Quebec or QC in any type of casing. Any other valid region (such as BC for British Columbia) returns a content domain sequence of EN_US|Global.

BE (Belgium)	Option_Language=FR
	Option_Language=NL*
		FR|Global
		NL|Global
			The content domain sequence is always assigned to FR|Global when Option_Language=FR. Any other specified language (such as NL for Dutch or EN_GB for British English), or if this option is blank, returns a content domain sequence of NL|Global.

CH (Switzerland)	Option_Language=FR
	Option_Language=IT
	Option_Language=RU*
		FR|Global
		IT|Global
		DE|Global
			The content domain sequence is always assigned to FR|Global or IT|Global when Option_Language=FR or IT, respectively. Any other specified language (such as RU for Russian), or if this option is

Option_Country input field	Additional input fields	Assigned content domain sequence	Notes
			blank, returns a content domain sequence of DE|Global.

i Note

*The content domain sequence is assigned independently from the output format.

See also:

- Reference Guide: Cleansing package options
- Reference Guide: Dynamic transform settings

Related Information

About output format [page 485]

2.16.2.9.2 About output format

Based on the specified domain in the output format, Data Cleanse uses certain output fields and formats the data in those fields according to the regional standards. You specify the domain in the *Output Format* of the Data Cleanse transform.

Based on regional standards, in some domains a compound given name is combined and output to the `Given Name1` field, while in other domains the first name is output to the `Given Name1` field and the second name is output to the `Given Name2` field.

Similarly, in some domains a compound family name is combined and output to the `Family Name1` field, while in other domains the first family name is output to the `Family Name1` field and the second family name is output to the `Family Name2` field.

In some domains the composite `Person` output field is comprised of the given name followed by the family name, while in other domains the composite `Person` output field is comprised of the family name followed by the given name.

The Data Cleanse transform requires that you specify an output format, even when your data is truly global.

When you have data from a single-domain region, specify the domain. For example, for Germany select *DE*, for China select *ZH*.

When you have data from a multi-domain region, you must select the preferred domain. Your data may be formatted differently depending on the domain you select.

For example, data from the Philippines may be output in English or Spanish output formats. As shown in the table below, the name **`Juan Carlos Sanchez Cruz`** will output in different fields depending on the selected output format.

Output field	Output format	
	English	Spanish
`Given Name1`	Juan	Juan
`Given Name2`	Carlos	Carlos
`Family Name1`	Sánchez Cruz	Sánchez
`Family Name2`		Cruz
`Person`	Juan Carlos Sánchez Cruz	Juan Carlos Sánchez Cruz

For Benelux data, you may choose to output your data in Dutch, French, or German output formats. As shown in the table below, the name **`H. D. BUDJHAWAN`** will output in different fields depending on the selected output format.

Output field	Output format		
	Dutch	French	German
`Given Name1`	H.D.	H. D.	H.
`Given Name2`			D.
`Family Name1`	Budjhawan	Budjhawan	Budjhawan
`Family Name2`			
`Person`	H.D. Budjhawan	H. D. Budjhawan	H. D. Budjhawan

You can modify the existing output format or add a new domain-specific output format by modifying the appropriate rules in your cleansing package.

Specifying the output format

You can set the output format in three different levels.

Level	Option
Default	Data Cleanse *Output format* transform option. This option is required. If there are errors or the intermediate-level and top-level options are not specified, this setting is used.
Intermediate	The data in Option_Country, Option_Language, and/or Option_Region input options. If there are errors or the top-level option is not specified, this setting is used. These input fields are populated and mapped from the Global Address Cleanse transform output fields ISO_Country_Code_2Char, Language, and Region1, respectively. Using these input fields is optional.
Top	The data in Option_Output_Format input option. The data in this option is assigned through a preceding Query transform or other preprocessing step. Using this input field is optional.

Example

During processing, Data Cleanse assigns the output format based on the top-level option, if the option is set and is valid.

Level	Option	Output	Notes
Top	Input field: Option_Output_Format=FR	The output format is French.	This option overrides the intermediate and default settings, unless this option is invalid (for example, misspelled) or not set.

Intermediate	Input fields:
	Option_Country=MX
	Option_Language=<not set>
	Option_Region=<not set>
		The output format is Spanish (local to Mexico).	This option overrides the default setting, unless this option contains invalid data, or was not set.
			In most cases, the output format can be identified based on the Option_Country input field only. However, there are certain countries that can use Option_Language or Option_Region to help determine the output format.

Default	Data Cleanse transform option: Output format=EN_US	The output format is English (local to United States).	This option is required and is used when there are errors in the top- and intermediate-level options.

When using the intermediate level, you will find that some countries can use additional input data to determine the output format.

Option_Country input field	Additional input fields	Assigned output format	Notes

CA (Canada)	Option_Region=QC
	Option_Region=BC*
		FR
		EN_US
			The output format is always assigned to FR when Option_Region is set to Quebec. Acceptable data includes Quebec or QC in any type of casing. Any other valid region (such as BC for British Columbia) returns an output format of EN_US.

BE (Belgium)	Option_Language=FR
	Option_Language=NL*
		FR
		NL
			The output format is always assigned to FR when Option_Language=FR. Any other specified language (such as NL for Dutch or EN_GB for British English), or

Option_Country input field	Additional input fields	Assigned output format	Notes
			if this option is blank, returns an output format of NL.

CH (Switzerland)	Option_Language=FR
	Option_Language=IT
	Option_Language=RU*
		FR
		IT
		DE
			The output format is always assigned to FR or IT when Option_Language=FR or IT, respectively. Any other specified language (such as RU for Russian), or if this option is blank, returns an output format of DE.

Note

*The output format is assigned independently of the content domain sequence.

Related Information

Reference Guide: Cleansing package options [page 1149]
Reference Guide: Dynamic transform settings [page 1129]

2.16.2.9.3 Customize prenames per country

When the input name does not include a prename, Data Cleanse generates the English prenames Mr. and Ms. To modify these terms, add a Query transform following the Data Cleanse transform and use the search_replace function to replace the terms with region-appropriate prenames.

2.16.2.9.4 Personal identification numbers

Data Cleanse can identify U.S. Social Security numbers and separate them into discrete components. If your data includes personal identification numbers other than U.S. Social Security numbers, you can create user-defined pattern rules to identify the numbers. User-defined pattern rules are part of the cleansing package and are defined in the *Edit Reference Data* tab of Cleansing Package Builder.

User-defined pattern rules are parsed in Data Cleanse with the *UDPM* parser. U.S. Social Security numbers are parsed in Data Cleanse with the *SSN* parser.

2.16.2.9.5 Text width in output fields

Many Japanese characters are represented in both fullwidth and halfwidth forms. Latin characters can be encoded in either a proportional or fullwidth form. In either case, the fullwidth form requires more space than the halfwidth or proportional form.

To standardize your data, you can use the *Character Width Style* option to set the character width for all output fields to either fullwidth or halfwidth. The normal width value reflects the normalized character width based on script type. Thus some output fields contain halfwidth characters and other fields contain fullwidth characters. For example, all fullwidth Latin characters are standardized to their halfwidth forms and all halfwidth katakana characters are standardized to their fullwidth forms. NORMAL_WIDTH does not require special processing and thus is the most efficient setting.

> Note
>
> Because the output width is based on the normalized width for the character type, the output data may be larger than the input data. You may need to increase the column width in the target table.

For template tables, selecting the *Use NVARCHAR for VARCHAR columns in supported databases box* changes the VARCHAR column type to NVARCHAR and allows for increased data size.

Related Information

Reference Guide: Locales and Multi-byte Functionality, Multi-byte support, Column Sizing [page 1808]

2.16.3 Geocoding

This section describes how the Geocoder transform works, different ways that you can use the transform, and how to understand your output.

> Note
>
> GeoCensus functionality in the USA Regulatory Address Cleanse transform will be deprecated in a future version of Data Services. It is recommended that you upgrade any data flows that currently use the GeoCensus functionality to use the Geocoder transform. For instructions on upgrading from GeoCensus to the Geocoder transform, see the *Upgrade Guide*.

How the Geocoder transform works

The Geocoder transform uses geographic coordinates expressed as latitude and longitude, addresses, and point-of-interest (POI) data. Using the transform, you can append addresses, latitude and longitude, census data, and other information to your data.

Based on mapped input fields, the Geocoder transform has three modes of geocode processing:

- Address geocoding
- Reverse geocoding
- POI textual search

Typically, the Geocoder transform is used in conjunction with the Global Address Cleanse or USA Regulatory Address Cleanse transform.

Related Information

Reference Guide: Transforms, Geocoder [page 1178]
Reference Guide: Data Quality Fields, Geocoder fields [page 1183]
Geocensus (USA Regulatory Address Cleanse) [page 671]

2.16.3.1 Address geocoding

In address geocoding mode, the Geocoder transform assigns geographic data. Based on the completeness of the input address data, the Geocoder transform can return multiple levels of latitude and longitude data. Appending different levels of latitude and longitude information to your data may help your organization target certain population sizes and other regional geographical data.

Prepare records for geocoding

The Geocoder transform works best when it has standardized and corrected address data, so to obtain the most accurate information you may want to place an address cleanse transform before the Geocoder transform in the work flow.

2.16.3.1.1 Latitude and longitude levels

Primary Number level

If your input data has a complete address (including the primary number), the Geocoder transform returns latitude and longitude coordinates to the exact location based on the directory type (range-based or parcel-based) that you subscribe to.

In general, the Geocoder transform uses geocoding directories to calculate latitude and longitude values for a house by interpolating between the beginning and ending point of a line segment, where the line segment represents a range of houses. The latitude and longitude values may be slightly offset from the exact location from where the house actually exists. This is called the primary range interpolated (PRI) assignment level.

Note

If you want to get an address-level assignment, the Primary_Number input field must be mapped and cannot be blank.

The Geocoder transform also supports parcel directories, which contain the most precise and accurate latitude and longitude values available for addresses, depending on the available country data. Parcel data is stored as points. Rather than getting you near the house, it takes you to the exact door. This is called the primary range exact (PRE) assignment level.

Postcode Centroid level

If an address has a postcode, you receive coordinates in the appropriate postcode area. The Geocoder transform has Postcode Full (PF), Postcode2 Partial (P2P) and Postcode1 (P1) assignment levels, depending on the available country data.

Locality Centroid level

If an address has a locality, you receive coordinates in the appropriate locality area. The Geocoder transform has Locality1 (L1), Locality2 (L2), Locality3 (L3) and Locality4 (L4) assignment levels, depending on the available country data.

2.16.3.1.2 Address geocoding field mapping

The following tables specify the input and output fields that are required, optional, or cannot be mapped in the address geocoding mode. The more input data you can provide, the better results you will obtain.

Input field mapping

Input field category	Address geocoding mode
Address	At least one required
Address POI	Optional
Latitude/Longitude	n/a
Max Records	n/a
Search Filter	n/a

Output field mapping

Output field category	Address geocoding mode
Address	n/a
Address POI	n/a
Assignment Level	Optional
Census	Optional
Distance	n/a
Info Code	Optional
Latitude/Longitude	Optional
Population	Optional
Results	n/a
Side of Street	Optional

2.16.3.1.3 Address geocoding scenario

Scenario: Use an address or an address and a point of interest to assign latitude and longitude information.

Number of output results: Single record

The following sections describe the required and optional input fields and available output fields to obtain results for this scenario. We also provide an example with sample data.

Required input fields

For required input fields, the Country field must be mapped. The more input data you can provide, the better results you will obtain.

Category	Input field name
Address	Country (required) Locality1–4 Postcode1–2 Primary_Name1–4 Primary_Number Primary_Postfix1

Category	Input field name
	Primary_Prefix1 Primary_Type1–4 Region1–2

Optional input fields

Category	Input field name
Address POI	POI_Name POI_Type

Available output fields

All output fields are optional.

Category	Output field name
Assignment Level	Assignment_Level Assignment_Level_Locality Assignment_Level_Postcode
Census	Census_Tract_Block Census_Tract_Block_Group Gov_County_Code Gov_Locality1_Code Gov_Region1_Code Metro_Stat_Area_Code Minor_Div_Code Stat_Area_Code
Info Code	Info_Code
Latitude/Longitude	Latitude Latitude_Locality Latitude_Postcode Latitude_Primary_Number

Category	Output field name
	Longitude Longitude_Locality Longitude_Postcode Longitude_Primary_Number
Population	Population_Class_Locality1
Side of Street	Side_Of_Primary_Address

Example

Input: You map input fields that contain the following data:

Input field name	Input value
Country	US
Postcode1	54601
Postcode2	4023
Primary_Name1	Front
Primary_Number	332
Primary_Type1	St.

Output: The mapped output fields contain the following results:

Output field name	Output value
Assignment_Level	PRE
Latitude	43.811616
Longitude	-91.256695

Related Information

Understanding your output [page 507]
Reference Guide: Data Quality fields, Geocoder fields, Input fields [page 1184]
Reference Guide: Data Quality fields, Geocoder fields, Output fields [page 1186]

2.16.3.2 Reverse geocoding

In reverse geocoding mode, the Geocoder transform identifies the closest address or point of interest based on an input reference location. Based on the input mapping, reverse geocoding can process in four modes:

- *Reverse with address input and single record output.*
  Use the address and optional address POI fields to determine a unique reference point, and output only the single closest record that matches the search filter.
  If search filter fields are not mapped, the Geocoder transform defaults to the nearest point of interest or address.
- *Reverse with address input and multiple record output.*
  Use the address and optional address POI fields to determine a unique reference point, and output the reference point data to output fields and multiple closest records that match the search filter fields to the Result_List field.
  The number of multiple records is determined by the Option_Max_Records input field if it is mapped and populated or the Max Records option.
- *Reverse with latitude/longitude input and single record output.*
  Use the latitude and longitude fields as the reference point, and output only the single closest record that matches the optional search filter fields.
  If the search filter fields are not mapped, the Geocoder transform defaults to the nearest point of interest or address.
- *Reverse with latitude/longitude input and multiple record output.*
  Use the latitude and longitude fields as the reference point, and output multiple closest records that match the search filter fields to the Result_List field.
  The number of multiple records is determined by the Option_Max_Records input field if it is mapped and populated or the Max Records option.
  If the search filter fields are not mapped, the Geocoder transform defaults to the nearest point of interest or address.

Note

- Mapping the Option_Radius input field lets you define the distance from the specified reference point and identify an area in which matching records are located.
- To find one or more locations that can be points of interest, addresses, or both, set the Search_Filter_Name or Search_Filter_Type input field. This limits the output matches to your search criteria.
- To return an address only, enter ADDR in the Search_Filter_Type input field.
- To return a point of interest only, enter the point-of-interest name or type.
- If you don't set a search filter, the transform returns both addresses and points of interest.

2.16.3.2.1 Reverse geocoding field mapping

The following tables specify the input and output fields that are required, optional, or cannot be mapped in the reverse geocoding mode. The more input data you can provide, the better results you will obtain.

Input field mapping

Input field category	Address input Single output	Address input Multiple output	Latitude/longitude input Single output	Latitude/longitude input Multiple output
Address	At least one required	At least one required	n/a	n/a
Address POI	Optional	Optional	n/a	n/a
Latitude/ Longitude	n/a	n/a	At least one required	At least one required
Max Records	n/a	At least one required	n/a	At least one required
Search Filter	At least one required	At least one required	Optional	Optional

Output field mapping

Output field category	Address input Single output	Address input Multiple output	Latitude/longitude input Single output	Latitude/longitude input Multiple output
Address	Optional	n/a	Optional	n/a
Address POI	Optional	n/a	Optional	n/a
Assignment Level	Optional	Optional	Optional	n/a
Census	Optional	n/a	Optional	n/a
Distance	Optional	n/a	Optional	n/a
Info Code	Optional	Optional	Optional	Optional
Latitude/ Longitude	Optional	Optional	Optional	n/a
Population	Optional	n/a	Optional	n/a
Results	n/a	Optional	n/a	Optional
Side of Street	Optional	Optional	Optional	n/a

2.16.3.2.2 Reverse geocoding scenarios

Reverse geocoding scenario 1

Scenario: Use latitude and longitude to find one or more addresses or points of interest.

The following sections describe the required and optional input fields and available output fields to obtain either single-record or multiple-record results for this scenario. We also provide an example with sample data.

Required input fields

For a single-record result, both Latitude and Longitude input fields must be mapped. For multiple-record results, the Latitude, Longitude, and Option_Max_Records input fields must all be mapped.

Category	Single-record results	Multiple-record results
	Input field name	Input field name
Latitude/Longitude	Latitude Longitude	Latitude Longitude
Max Records	n/a	Option_Max_Records

Optional input fields

Category	Single-record results	Multiple-record results
	Input field name	Input field name
Search Filter	Option Radius Search_Filter_Name Search_Filter_Type	Option_Radius Search_Filter_Name Search_Filter_Type

Available output fields

All output fields are optional.

Category	Single-record results	Multiple-record results
	Output field name	Output field name
Address	Address_Line Country_Code Locality1–4 Postcode Postcode1–2 Primary_Name1–4 Primary_Number Primary_Postfix1	n/a

Category	Single-record results	Multiple-record results
	Output field name	Output field name
	Primary_Prefix1 Primary_Range_High Primary_Range_Low Primary_Type1–4 Region1–2	
Address POI	POI_Name POI_Type	n/a
Assignment Level	Assignment_Level Assignment_Level_Locality Assignment_Level_Postcode	n/a
Census	Census_Tract_Block Census_Tract_Block_Group Gov_County_Code Gov_Locality1_Code Gov_Region1_Code Metro_Stat_Area_Code Minor_Div_Code Stat_Area_Code	n/a
Distance	Distance	n/a
Info Code	Info_Code	Info_Code
Latitude/Longitude	Latitude Latitude_Locality Latitude_Postcode Latitude_Primary_Number Longitude Longitude_Locality Longitude_Postcode Longitude_Primary_Number	n/a
Population	Population_Class_Locality1	n/a

Category	Single-record results Output field name	Multiple-record results Output field name
Results	n/a	Result_List Result_List_Count
Side of Street	Side_Of_Primary_Address	n/a

Example

The following example illustrates a scenario using latitude and longitude and a search filter to output a single point of interest closest to the input latitude and longitude.

Input: You map input fields that contain the following data:

Input field name	Input value
Latitude	43.811616
Longitude	-91.256695
Search_Filter_Name	BUSINESS OBJECTS

Output: The mapped output fields contain the following results:

Output field name	Output value
Address_Line	332 FRONT ST
Assignment_Level	PRE
Country_Code	US
Distance	1.3452
Locality1	LA CROSSE
Postcode	54601-4023
Postcode1	54601
Postcode2	4023
Primary_Name1	FRONT
Primary_Number	332
Primary_Type1	ST
POI_Name	BUSINESS OBJECTS
POI_Type	5800
Region1	WI

Reverse geocoding scenario 2

Scenario: Use an address or point of interest to find one or more closest addresses or points of interest. In addition, the Geocoder transform outputs latitude and longitude information for both the input reference point and the matching output results.

The following sections describe the required and optional input fields and available output fields to obtain either single-record or multiple-record results for this scenario. We also provide examples with sample data.

Required input fields

For required input fields, at least one input field in each category must be mapped.

Category	Single-record results	Multiple-record results
	Input field name	Input field name
Address	Address_Line Country Locality1–4 Postcode Postcode1–2 Primary_Name1–4 Primary_Number Primary_Postfix1 Primary_Prefix1 Primary_Range_High Primary_Range_Low Primary_Type1–4 Region1–2	Latitude Longitude
Max Records	n/a	Option_Max_Records
Search Filter	Option_Radius Search_Filter_Name Search_Filter_Type	Option_Radius Search_Filter_Name Search_Filter_Type

Optional input fields

Category	Single-record results Input field name	Multiple-record results Input field name
Address POI	POI_Name POI_Type	POI_Name POI_Type

Available output fields

All output fields are optional.

For a single-record result, the output fields are the results for the spatial search.

For multiple-record results, the output fields in the Assignment Level and Latitude/Longitude categories are the results for the reference address assignment. Output fields in the Results category are the results for the reverse geocoding.

Category	Single-record results Output field name	Multiple-record results Output field name
Address	Address_Line Country_Code Locality1–4 Postcode Postcode1–2 Primary_Name1–4 Primary_Number Primary_Postfix1 Primary_Prefix1 Primary_Range_High Primary_Range_Low Primary_Type1–4 Region1–2	n/a
Address POI	POI_Name POI_Type	n/a
Assignment Level	Assignment_Level	Assignment_Level

Category	Single-record results Output field name	Multiple-record results Output field name
	Assignment_Level_Locality Assignment_Level_Postcode	Assignment_Level_Locality Assignment_Level_Postcode
Census	Census_Tract_Block Census_Tract_Block_Group Gov_County_Code Gov_Locality1_Code Gov_Region1_Code Metro_Stat_Area_Code Minor_Div_Code Stat_Area_Code	n/a
Distance	Distance	n/a
Info Code	Info_Code	Info_Code
Latitude/Longitude	Latitude Latitude_Locality Latitude_Postcode Latitude_Primary_Number Longitude Longitude_Locality Longitude_Postcode Longitude_Primary_Number	Latitude Latitude_Locality Latitude_Postcode Latitude_Primary_Number Longitude Longitude_Locality Longitude_Postcode Longitude_Primary_Number
Population	Population_Class_Locality1	n/a
Results	n/a	Result_List Result_List_Count
Side of Street	Side_Of_Primary_Address	Side_Of_Primary_Address

Example

The following example illustrates a scenario using an address and a search filter to output a single point of interest closest to the input address. The request is to find the closest bank (POI type 6000) to the input location. The transform also outputs latitude and longitude information for the output result.

Input: You map input fields that contain the following data.

Input field name	Input value
Country	US
Postcode1	55601
Postcode2	4023
Primary_Name1	Front
Primary_Number	332
Primary_Type1	St.
POI_Name	BUSINESS OBJECTS
Search_Filter_Type	6000

Output: The mapped output fields contain the following results:

Output field name	Output value
Address_Line	201 MAIN ST
Assignment_Level	PRE
Country_Code	US
Distance	0.4180
Locality1	LA CROSSE
POI_Name	US BANK
POI_Type	6000
Postcode1	54601
Primary_Name1	MAIN
Primary_Number	201
Primary_Type1	ST
Region1	WI

Related Information

Understanding your output [page 507]
Reference Guide: Data Quality fields, Geocoder fields, Input fields [page 1184]
Reference Guide: Data Quality fields, Geocoder fields, Output fields [page 1186]

2.16.3.3 POI textual search

In the POI textual search mode, the Geocoder transform uses address fields and POI name or type fields as search criteria to match with points of interest. The results are output in the Result_List XML output field.

The number of multiple records is determined by the Option_Max_Records input field if it is mapped and populated or the Max Records option.

2.16.3.3.1 POI textual search field mapping

The following tables specify the input and output fields that are required, optional, or cannot be mapped in the POI textual search mode. The more input data you can provide, the better results you will obtain.

Input field mapping

In the POI textual search mode, at least one input field in the address POI category must be mapped, which is the key difference between the POI textual search and address geocoding modes.

Input field category	Address geocoding mode
Address	At least one required
Address POI	At least one required
Latitude/Longitude	n/a
Max Records	At least one required
Search Filter	n/a

Output field mapping

Output field category	Address geocoding mode
Address	n/a
Address POI	n/a
Assignment Level	n/a
Census	n/a
Distance	n/a
Info Code	Optional
Latitude/Longitude	n/a
Population	n/a
Results	Optional
Side of Street	n/a

2.16.3.3.2 POI textual search scenario

Scenario: Use an address and point-of-interest information to identify a list of potential point-of-interest matches.

Number of output results: Multiple records. The number of records is determined by the Option_Max_Records input field (if populated), or the Max Records option.

The following sections describe the required input fields and available output fields to obtain results for this scenario. We also provide an example with sample data.

Required input fields

For required input fields, at least one input field in each category must be mapped. The Country field must be mapped. The more input data you can provide, the better results you will obtain.

Category	Input field name
Address	Country (required) Locality1–4 Postcode1–2 Primary_Name1–4 Primary_Number Primary_Postfix1 Primary_Prefix1 Primary_Type1–4 Region1–2
Address POI	POI_Name POI_Type
Max Records	Option_Max_Records

Optional input fields

Not applicable.

Available output fields

All output fields are optional.

Category	Output field name
Info Code	Info_Code
Results	Result_List Result_List_Count

Example

The following example illustrates a scenario using POI textual search to identify a list of potential point-of-interest matches (all “BUSINESS OBJECTS” records on Front Street).

Input: You map input fields that contain the following data:

Input field name	Input value
Country	US
Postcode1	54601
Postcode2	4023
Primary_Name1	Front
Primary_Type1	St.
POI_Name	BUSINESS OBJECTS
Option_Max_Records	10

Output: The mapped output fields contain the following results with one record:

Output field name	Output value
Result_List	Output as XML; example shown below
Result_List_Count	1

Result_List XML: The XML result for this example has one record.

```
<RESULT_LIST>
  <RECORD>
    <ADDRESS_LINE>332 FRONT ST</ADDRESS_LINE>
    <ASSIGNMENT_LEVEL>PRE</ASSIGNMENT_LEVEL>
    <COUNTRY_CODE>US</COUNTRY_CODE>
    <DISTANCE>0.3340</DISTANCE>
    <LATITUDE>43.811616</LATITUDE>
    <LOCALITY1>LA CROSSE</LOCALITY1>
    <LONGITUDE>-91.256695</LONGITUDE>
    <POI_NAME>BUSINESS OBJECTS</POI_NAME>
    <POI_TYPE>5800</POI_TYPE>
    <POSTCODE>56001-4023</POSTCODE>
    <POSTCODE1>56001</POSTCODE1>
    <POSTCODE2>4023</POSTCODE2>
    <PRIMARY_NAME1>FRONT</PRIMARY_NAME1>
    <PRIMARY_NUMBER>332</PRIMARY_NUMBER>
    <PRIMARY_TYPE1>ST</PRIMARY_TYPE1>
    <RANKING>1</RANKING>
    <REGION1>WI</REGION1>
  </RECORD>
</RESULT_LIST>
```

2.16.3.4 Understanding your output

Latitude and longitude

On output from the Geocoder transform, you will have latitude and longitude data. Latitude and longitude are denoted on output by decimal degrees; for example, 12.12345. Latitude (0-90 degrees north or south of the equator) shows a negative sign in front of the output number when the location is south of the equator. Longitude (0-180 degrees east or west of Greenwich Meridian in London, England) shows a negative sign in front of the output number when the location is within 180 degrees west of Greenwich.

Assignment level

You can understand the accuracy of the assignment based on the Assignment_Level output field. The return code of PRE means that you have the finest depth of assignment available to the exact location. The second finest depth of assignment is a return code of PRI, which is the primary address range, or house number. The most general output level is either P1 (Postcode level) or L1 (Locality level), depending on the value that you chose in the Best Assignment Level option.

Multiple results

For multiple-record results, the Result_List output field is output as XML which can contain the following output, depending on the available data.

Category	Output field name
Address	Address_Line Country_Code Locality1–4 Postcode Postcode1–2 Primary_Name1–4 Primary_Number Primary_Postfix1 Primary_Prefix1

Category	Output field name
	Primary_Range_High Primary_Range_Low Primary_Type1–4 Region1–2
Address POI	POI_Name POI_Type
Assignment Level	Assignment_Level
Distance	Distance
Latitude/Longitude	Latitude Longitude
Ranking	Ranking

Standardize address information

The geocoding data provided by vendors is not standardized. To standardize the address data that is output by the Geocoder transform, you can insert a Global Address Cleanse or USA Regulatory Address Cleanse transform in the data flow after the Geocoder transform. If you have set up the Geocoder transform to output multiple records, the address information in the XML output string must first be unnested before it can be cleansed.

Related Information

Reference Guide: Transforms, Geocoder options [page 1180]

2.16.3.5 Working with other transforms

Typically, the Geocoder transform is used in conjunction with the Global Address Cleanse or USA Regulatory Address Cleanse transform.

Global Address Cleanse transform

For the Geocoder transform to provide the most accurate information, make sure the Global Address Cleanse transform output fields are mapped to Geocoder transform input fields as follows:

Global Address Cleanse output	Geocoder input field
ISO_Country_Code_2Char	Country
Postcode1	Postcode1
Postcode2	Postcode2
Primary_Name1	Primary_Name1
Primary_Number	Primary_Number
Primary_Postfix1	Primary_Postfix1
Primary_Prefix1	Primary_Prefix1
Primary_Type1	Primary_Type1
Region1_Symbol	Locality1

It is also recommended that the Global Address Cleanse transform standardization options are set as follows:

Global Address Cleanse standardization option	Value
Assign Locality	Convert
Directional Style	Short
Primary Type Style	Short

➜ Tip

Postal address versus administrative address: The Global Address Cleanse transform uses postal addresses, while the Geocoder transform venders, NAVTEQ and TOMTOM, use administrative addresses. Although a postal alignment is performed with postal addresses, there are some gaps. For example, for one Canadian address, the locality from the Global Address Cleanse transform might return TORONTO, but the Geocoder directory stores NORTH YORK. In this case, when the Locality1 field is mapped on input, it results in an Address Not Found error. As a workaround for this issue, you can remove the Locality1 input field mapping to get a better geocoding assignment.

2.16.4 Match

2.16.4.1 Match and consolidation using Data Services and Information Steward

When manual review is necessary, use SAP Data Services to automatically implement the overall matching and consolidation process. This process stages the match groups for manual review in SAP Information Steward.

To implement this process, consider the following factors:

- Do you stage only suspect match groups (groups that do not meet a match score or threshold that you define) for manual review? Or do you stage all the match results including unique records for match review and let data stewards manually move suspect records or unique records to the group they should belong? Note that the target group in this case could be a suspect group or “high confidence” group.

- Do you want to enable best record creation within Information Steward? Or do you want the Information Steward manual review to only modify suspect match groups as necessary and have Data Services handle the best record creation after the manual review is complete?
- If you want to enable best record creation within Information Steward, when do you perform the best record creation for high confidence match groups?
  - Manually creating a best record for all match groups in a large data set might not be practical.
  - On the other hand, if you have already consolidated high-confidence match groups within Data Services, then you have already modified the original match group to include the best record values, and you cannot stage these modified match groups for manual review in Information Steward.

Consider the following approaches for match review:

- Engage data stewards and domain experts in manual review of suspect match groups only. They can modify the suspect match groups if necessary and create the best record for these match groups.
  1. The Data Services job performs automated matching, identifies suspect match groups, and populates only suspect match groups in a staging database. In this scenario, “high confidence match groups” must be separated from “suspect match groups”. The best record can be created for the “high confidence match groups” and those match groups can be consolidated within Data Services.
     This process can be accomplished by routing the “high confidence match groups” to another Match transform that is configured to create best records out of already matched groups of records.
  2. Data stewards and domain experts review the “suspect match groups” in Information Steward and make corrections as appropriate (for example unmatch or move a record). They can manually create the best record for these match groups.
- Engage data stewards and domain experts in manual review and correction of all match groups, but keep the best record creation and consolidation logic in Data Services.
  1. The Data Services job performs automated matching and populates all match results into the staging database. A match result table in the staging database contains suspect match groups, “high confidence” match groups, and unique records.
  2. Data stewards and domain experts review the suspect match groups in Information Steward and make corrections as appropriate. They can move the records between suspect match groups and “high confidence” match groups, or include a unique record in the appropriate match group.
  3. After the match review is finished, all match groups are processed in another Data Services job which creates the best record and consolidates all match groups.
     This process could be accomplished by a Match transform that is configured to create best records out of already matched groups of records.

> **Note**
>
> You can find examples of best record creation using Information Steward or Data Services by downloading the Data Quality Management Match blueprints. You can find more information about blueprints at *http://scn.sap.com/docs/DOC-8820*.

2.16.4.2 Matching strategies

Here are a few examples of strategies to help you think about how you want to approach the setup of your matching data flow.

Strategy	Description
Simple match	Use this strategy when your matching business rules consist of a single match criteria for identifying relationships in consumer, business, or product data.
Consumer Householding	Use this strategy when your matching business rules consist of multiple levels of consumer relationships, such as residential matches, family matches, and individual matches.
Corporate Householding	Use this strategy when your matching business rules consist of multiple levels of corporate relationships, such as corporate matches, subsidiary matches, and contact matches.
Multinational consumer match	Use this match strategy when your data consists of multiple countries and your matching business rules are different for different countries
Identify a person multiple ways	Use this strategy when your matching business rules consist of multiple match criteria for identifying relationships, and you want to find the overlap between all of those definitions.

Think about the answers to these questions before deciding on a match strategy:

- What does my data consist of? (Customer data, international data, and so on)
- What fields do I want to compare? (last name, firm, and so on.)
- What are the relative strengths and weaknesses of the data in those fields?

  > ➜ Tip
  >
  > You will get better results if you cleanse your data before matching. Also, data profiling can help you answer this question.

- What end result do I want when the match job is complete? (One record per family, per firm, and so on.)

Related Information

Association matching [page 579]

2.16.4.3 Match components

The basic components of matching are:

- Match sets
- Match levels
- Match criteria

Match sets

A match set is represented by a Match transform on your workspace. Each match set can have its own break groups, match criteria, and prioritization.

A match set has three purposes:

- To allow only select data into a given set of match criteria for possible comparison (for example, exclude blank SSNs, international addresses, and so on).
- To allow for related match scenarios to be stacked to create a multi-level match set.
- To allow for multiple match sets to be considered for association in an Associate match set.

Match levels

A match level is an indicator to what type of matching will occur, such as on individual, family, resident, firm, and so on. A match level refers not to a specific criteria, but to the broad category of matching.

You can have as many match levels as you want. However, the Match wizard restricts you to three levels during setup (more can be added later). You can define each match level in a match set in a way that is increasingly more strict. Multi-level matching feeds only the records that match from match level to match level (for example, resident, family, individual) for comparison.

Match component	Description
Family	The purpose of the family match type is to determine whether two people should be considered members of the same family, as reflected by their record data. The Match transform compares the last name and the address data. A match means that the two records represent members of the same family. The result of the match is one record per family.
Individual	The purpose of the individual match type is to determine whether two records are for the same person, as reflected by their record data. The Match transform compares the first name, last name, and address data. A match means that the two records represent the same person. The result of the match is one record per individual.
Resident	The purpose of the resident match type is to determine whether two records should be considered members of the same residence, as reflected by their record data. The Match transform compares the address data. A match means that the two records represent members of the same household. Contrast this match type with the family match type, which also compares last-name data. The result of the match is one record per residence.
Firm	The purpose of the firm match type is to determine whether two records reflect the same firm. This match type involves comparisons of firm and address data. A match means that the two records represent the same firm. The result of the match is one record per firm.

Match component	Description
Firm-Individual	The purpose of the firm-individual match type is to determine whether two records are for the same person at the same firm, as reflected by their record data. With this match type, we compare the first name, last name, firm name, and address data. A match means that the two records reflect the same person at the same firm. The result of the match is one record per individual per firm.

Match criteria

Match criteria refers to the field you want to match on. You can use criteria options to specify business rules for matching on each of these fields. They allow you to control how close to exact the data needs to be for that data to be considered a match.

For example, you may require first names to be at least 85% similar, but also allow a first name initial to match a spelled out first name, and allow a first name to match a middle name.

- Family level match criteria may include family (last) name and address, or family (last) name and telephone number.
- Individual level match criteria may include full name and address, full name and SSN, or full name and e-mail address.
- Firm level match criteria may include firm name and address, firm name and Standard Industrial Classification (SIC) Code, or firm name and Data Universal Numbering System (DUNS) number.

2.16.4.4 Match Wizard

2.16.4.4.1 Match wizard

The Match wizard can quickly set up match data flows, without requiring you to manually create each individual transform it takes to complete the task.

What the Match wizard does

The Match wizard:

- Builds all the necessary transforms to perform the match strategy you choose.
- Applies default values to your match criteria based on the strategy you choose.
- Places the resulting transforms on the workspace, connected to the upstream transform you choose.
- Detects the appropriate upstream fields and maps to them automatically.

What the Match wizard does not do

The Match wizard provides you with a basic match setup that in some cases, will require customization to meet your business rules.

The Match wizard:

- Does not alter any data that flows through it. To correct non-standard entries or missing data, place one of the address cleansing transforms and a Data Cleanse transform upstream from the matching process.
- Does not connect the generated match transforms to any downstream transform, such as a Loader. You are responsible for connecting these transforms.
- Does not allow you to set rule-based or weighted scoring values for matching. The Match wizard incorporates a "best practices" standard that set these values for you. You may want to edit option values to conform to your business rules.

Related Information

Combination method [page 551]

2.16.4.4.2 Before you begin

Prepare a data flow for the Match wizard

To maximize its usefulness, be sure to include the following in your data flow before you launch the Match wizard:

- Include a Reader in your data flow. You may want to match on a particular input field that our data cleansing transforms do not handle.
- Include one of the address cleansing transforms and the Data Cleanse transform. The Match wizard works best if the data you're matching has already been cleansed and parsed into discrete fields upstream in the data flow.
- If you want to match on any address fields, be sure that you pass them through the Data Cleanse transform. Otherwise, they will not be available to the Match transform (and Match Wizard). This rule is also true if you have the Data Cleanse transform before an address cleanse transform.

2.16.4.4.3 Use the Match Wizard

2.16.4.4.3.1 Select match strategy

The Match wizard begins by prompting you to choose a match strategy, based on your business rule requirements. The path through the Match wizard depends on the strategy you select here. Use these descriptions to help you decide which strategy is best for you:

- **Simple match.** Use this strategy when your matching business rules consist of a single match criteria for identifying relationships in consumer, business, or product data.
- **Consumer Householding.** Use this strategy when your matching business rules consist of multiple levels of consumer relationships, such as residential matches, family matches, and individual matches.
- **Corporate Householding.** Use this strategy when your matching business rules consist of multiple levels of corporate relationships, such as corporate matches, subsidiary matches, and contact matches.
- **Multinational consumer match.** Use this match strategy when your data consists of multiple countries and your matching business rules are different for different countries.

  > Note
  >
  > The multinational consumer match strategy sets up a data flow that expects Latin1 data. If you want to use Unicode matching, you must edit your data flow after it has been created.
- **Identify a person multiple ways.** Use this strategy when your matching business rules consist of multiple match criteria for identifying relationships, and you want to find the overlap between all of those definitions.

Source statistics

If you want to generate source statistics for reports, make sure a field that houses the physical source value exists in all of the data sources.

To generate source statistics for your match reports, select the *Generate statistics for your sources* checkbox, and then select a field that contains your physical source value.

Related Information

Unicode matching [page 580]

Association matching [page 579]

2.16.4.4.3.2 Identify matching criteria

Criteria represent the data that you want to use to help determine matches. In this window, you will define these criteria for each match set that you are using.

Match sets compare data to find similar records, working independently within each break group that you designate (later in the Match wizard). The records in one break group are not compared against those in any other break group.

To find the data that matches all the fields, use a single match set with multiple fields. To find the data that matches only in a specific combination of fields, use multiple match sets with two fields.

When working on student or snowbird data, an individual may use the same name but have multiple valid addresses.

Select a combination of fields that best shows which information overlaps, such as the family name and the SSN.

Data1	Data2	Data3	Data4
R. Carson	1239 Whistle Lane	Columbus, Ohio	555-23-4333
Robert T. Carson	52 Sunbird Suites	Tampa, Florida	555-23-4333

1. Enter the number of ways you have to identify an individual. This produces the corresponding number of match sets (transforms) in the data flow.
2. The default match set name appears in the *Name* field. Select a match set in the *Match sets* list, and enter a more descriptive name if necessary.
3. For each match set, choose the criteria you want to match on.

   Later, you will assign fields from upstream transforms to these criteria.
4. Select the option you want to use for comparison in the Compare using column. The options vary depending on the criteria chosen. The compare options are:
   - *Field similarity*
   - *Word similarity*
   - *Numeric difference*
   - *Numeric percent difference*
   - *Geo proximity*
5. Optional: If you choose to match on Custom, enter a name for the custom criteria in the Custom name column.
6. Optional: If you choose to match on Custom, specify how close the data must be for that criteria in two records to be considered a match. The values that result determine how similar you expect the data to be during the comparison process for this criteria only. After selecting a strategy, you may change the values for any of the comparison rules options in order to meet your specific matching requirements.Select one of the following from the list in the Custom exactness column:
   - *Exact*: Data in this criteria must be exactly the same; no variation in the data is allowed.
   - *Tight*: Data in this criteria must have a high level of similarity; a small amount of variation in the data is allowed.
   - *Medium*: Data in this criteria may have a medium level of similarity; a medium amount of variation in the data is allowed.
   - *Loose*: Data in this criteria may have a lower level of similarity; a greater amount of variation in the data is allowed.

2.16.4.4.3.3 Define match levels

Match levels allow matching processes to be defined at distinct levels that are logically related. Match levels refer to the broad category of matching not the specific rules of matching. For instance, a residence-level match would match on only address elements, a family-level would match on only Last Name and then the individual-level would match on First Name.

Multi-level matching can contain up to 3 levels within a single match set defined in a way that is increasingly more strict. Multi-level matching feeds only the records that match from match level to match level (that is, resident, family, individual) for comparison.

To define match levels:

1. Click the top level match, and enter a name for the level, if you don't want to keep the default name. The default criteria is already selected. If you do not want to use the default criteria, click to remove the check mark from the box.

   The default criteria selection is a good place to start when choosing criteria. You can add criteria for each level to help make finer or more precise matches.
2. Select any additional criteria for this level.
3. If you want to use criteria other than those offered, click *Custom* and then select the desired criteria.
4. Continue until you have populated all the levels that you require.

2.16.4.4.3.4 Select countries

Select the countries whose postal standards may be required to effectively compare the incoming data. The left panel shows a list of all available countries. The right panel shows the countries you already selected.

1. Select the country name in the All Countries list.
2. Click *Add* to move it into the Selected Countries list.
3. Repeat steps 1 and 2 for each country that you want to include.

   You can also select multiple countries and add them all by clicking the *Add* button.

The countries that you select are remembered for the next Match wizard session.

2.16.4.4.3.5 Group countries into tracks

Create tracks to group countries into logical combinations based on your business rules (for example Asia, Europe, South America). Each track creates up to six match sets (Match transforms).

1. Select the number of tracks that you want to create. The *Tracks* list reflects the number of tracks you choose and assigns a track number for each.
2. To create each track, select a track title, such as Track1.
3. Select the countries that you want in that track.
4. Click *Add* to move the selected countries to the selected track.

   Use the COUNTRY UNKNOWN (__) listing for data where the country of origin has not been identified.

   Use the COUNTRY OTHER (--) listing for data whose country of origin has been identified, but the country does not exist in the list of selected countries.
5. From Match engines, select one of the following engines for each track:

   > Note
   >
   > All match transforms generated for the track will use the selected Match engine.

   - *LATIN1* (Default)
   - *CHINESE*
   - *JAPANESE*

- *KOREAN*
- *TAIWANESE*
- *OTHER_NON_LATIN1*

The *Next* button is only enabled when all tracks have an entry and all countries are assigned to a track.

2.16.4.4.3.6 Select criteria fields

Select and deselect criteria fields for each match set and match level you create in your data flow. These selections determine which fields are compared for each record. Some criteria may be selected by default, based on the data input.

If there is only one field of the appropriate content type, you will not be able to change the field for that criteria within the Match Wizard.

To enable the *Next* button, you must select at least one non-match-standard field.

1. For each of the criteria fields you want to include, select an available field from the drop-down list, which contains fields from upstream source(s). The available fields are limited to the appropriate content types for that criteria. If no fields of the appropriate type are available, all upstream fields display in the menu.
2. Optional: Deselect any criteria fields you do not want to include.

2.16.4.4.3.7 Create break keys

Use break keys to create manageable groups of data to compare. The match set compares the data in the records within each break group only, not across the groups. Making the correct selections can save valuable processing time by preventing widely divergent data from being compared.

Break keys are especially important when you deal with large amounts of data, because the size of the break groups can affect processing time. Even if your data is not extensive, break groups will help to speed up processing.

Create break keys that group similar data that would most likely contain matches. Keep in mind that records in one break group will not be compared against records in any other break group.

For example, when you match to find duplicate addresses, base the break key on the postcode, city, or state to create groups with the most likely matches. When you match to find duplicate individuals, base the break key on the postcode and a portion of the name as the most likely point of match.

To create break keys:

1. In the How many fields column, select the number of fields to include in the break key.
2. For each break key, select the following:
    - the field(s) in the break key
    - the starting point for each field
    - the number of positions to read from each field
3. After you define the break keys, do one of the following:

- Click *Finish*. This completes the match transform.
- If you are performing multi-national matching, click *Next* to go to the Matching Criteria page.

2.16.4.4.4 After setup

Although the Match wizard does a lot of the work, there are some things that you must do to have a runnable match job. There are also some things you want to do to refine your matching process.

Connect to downstream transforms

When the Match wizard is complete, it places the generated transforms on the workspace, connected to the upstream transform you selected to start the Match wizard. For your job to run, you must connect each port from the last transform to a downstream transform. To do this, click a port and drag to connect to the desired object.

View and edit the new match transform

To see what is incorporated in the transform(s) the Match Wizard produces, right-click the transform and choose *Match Editor*.

View and edit Associate transforms

To see what is incorporated in the Associate transform(s) the Match Wizard produces, right-click the transform and choose *Associate Editor*.

Multinational matching

For the Multinational consumer match strategy, the wizard builds a Match transform for each track that you create.

> ⚠ Caution
>
> If you delete any tracks from the workspace after the wizard builds them, you must open the Case transform and delete any unwanted rules.

Related Information

Unicode matching [page 580]

2.16.4.5 Transforms for match data flows

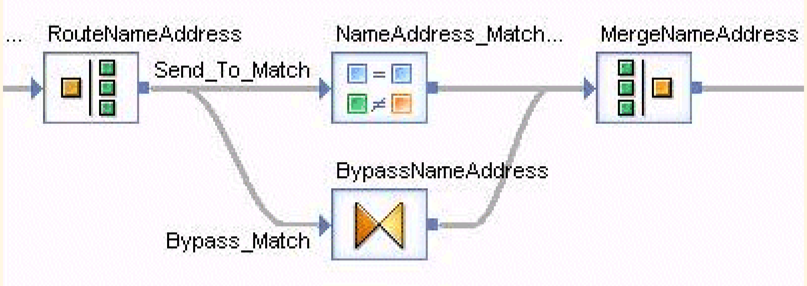
The Match and Associate transforms are the primary transforms involved in setting up matching in a data flow. These transforms perform the basic matching functions.

There are also other transforms that can be used for specific purposes to optimize matching.

Transform	Usage
Case	Routes data to a particular Match transform (match set). A common usage for this transform is to send USA-specific and international-specific data to different transforms. You can also use this transform to route blank records around a Match transform.
Merge	Performs the following functions: • Brings together data from Match transforms for Association matching. • Brings together matching records and blank records after being split by a Case transform.
Query	Creates fields, performs functions to help prepare data for matching, orders data, and so on.

Example

Any time you need to bypass records from a particular match process (usually in Associative data flows and any time you want to have records with blank data to bypass a match process) you will use the Case, Query, and Merge transforms.



- The Case transform has two routes: one route sends all records that meet the criteria to the Match transform, and one that sends all other records to the bypass match route.
- The Query transform adds the fields that the Match transform generates and that you output. (The output schema in the Match transform and the output schema in the Query transform must be identical for the routes to be merged.) The contents of the newly added fields in the Query transform may be populated with an empty string.
- The Merge transform merges the two routes into a single route.

2.16.4.5.1 To remove matching from the Match transform

You may want to place a transform that employs some of the functionality of a Match transform in your data flow, but does not include the actual matching features. For example, you may want to do candidate selection or prioritization in a data flow or a location in a data flow.

1. Right-click the Match transform in the object library, and choose New.
2. In the *Format name* field, enter a meaningful name for your transform. It's helpful to indicate which type of function this transform will be performing.
3. Click *OK*.
4. Drag and drop your new Match transform configuration onto the workspace and connect it to your data flow.
5. Right-click the new transform, and choose *Match Editor*.
6. Deselect the *Perform matching* option in the upper left corner of the Match editor.

Now you can add any available operation to this transform.

2.16.4.6 Working in the Match and Associate editors

Editors

The Match and Associate transform editors allow you to set up your input and output schemas. You can access these editors by double-clicking the appropriate transform icon on your workspace.

The Match and Associate editors allow you to configure your transform's options. You can access these editors by right-clicking the appropriate transform and choosing *Match Editor* (or *Associate Editor*).

Order of setup

> ➔ Tip
>
> The order that you set up your Match transform is important!

First, it is best to map your input fields. If you don't, and you add an operation in the Match editor, you may not see a particular field you want to use for that operation.

Secondly, you should configure your options in the Match editor before you map your output fields. Adding operations to the Match transform (such as Unique ID and Group Statistics) can provide you with useful Match transform-generated fields that you may want to use later in the data flow or add to your database.

> ➔ Remember
>
> Make sure that you:
>
> 1. Map your input fields.
> 2. Configure the options for the transform.

3. Map your output fields.

2.16.4.7 Physical and logical sources

Tracking your input data sources and other sources, whether based on an input source or based on some data element in the rows being read, throughout the data flow is essential for producing informative match reports. Depending on what you are tracking, you must create the appropriate fields in your data flow to ensure that the software generates the statistics you want, if you don't already have them in your database.

- Physical source: The filename or value attributed to the source of the input data.
- Logical source: A group of records spanning multiple input sources or a subset of records from a single input source.

Physical input sources

You track your input data source by assigning that physical source a value in a field. Then you will use this field in the transforms where report statistics are generated.

To assign this value, add a Query transform after the source and add a column with a constant containing the name you want to assign to this source.

i Note

If your source is a flat file, you can use the *Include file name* option to automatically generate a column containing the file name.

Logical input sources

If you want to count source statistics in the Match transform (for the Match Source Statistics Summary report, for example), you must create a field using a Query transform or a User-Defined transform, if you don't already have one in your input data sources.

This field tracks the various sources within a Reader for reporting purposes, and is used in the Group Statistics operation of the Match transform to generate the source statistics. It is also used in compare tables, so that you can specify which sources to compare.

2.16.4.7.1 Using sources

A source is the grouping of records on the basis of some data characteristic that you can identify. A source might be all records from one input file, or all records that contain a particular value in a particular field.

Sources are abstract and arbitrary—there is no physical boundary line between sources. Source membership can cut across input files or database records as well as distinguish among records within a file or database, based on how you define the source.

If you are willing to treat all your input records as normal, eligible records with equal priority, then you do not need to include sources in your job.

Typically, a match user expects some characteristic or combination of characteristics to be significant, either for selecting the best matching record, or for deciding which records to include or exclude from a mailing list, for example. Sources enable you to attach those characteristics to a record, by virtue of that record’s membership in its particular source.

Before getting to the details about how to set up and use sources, here are some of the many reasons you might want to include sources in your job:

- To give one set of records priority over others. For example, you might want to give the records of your house database or a suppression source priority over the records from an update file.
- To identify a set of records that match suppression sources, such as the DMA.
- To set up a set of records that should not be counted toward multi-source status. For example, some mailers use a seed source of potential buyers who report back to the mailer when they receive a mail piece so that the mailer can measure delivery. These are special-type records.
- To save processing time, by canceling the comparison within a set of records that you know contains no matching records. In this case, you must know that there are no matching records within the source, but there may be matches among sources. To save processing time, you could set up sources and cancel comparing within each source.
- To get separate report statistics for a set of records within an source, or to get report statistics for groups of sources.
- To protect a source from having its data overwritten by a best record or unique ID operation. You can choose to protect data based on membership in a source.

2.16.4.7.2 Source types

You can identify each source as one of three different types: Normal, Suppression, or Special. The software can process your records differently depending on their source type.

Source	Description
Normal	A Normal source is a group of records considered to be good, eligible records.
Suppress	A Suppress source contains records that would often disqualify a record from use. For example, if you’re using Match to refine a mailing source, a suppress source can help remove records from the mailing. Examples: • DMA Mail Preference File • American Correctional Association prisons/jails sources • No pandering or non-responder sources • Credit card or bad-check suppression sources

Source	Description
Special	A Special source is treated like a Normal source, with one exception. A Special source is not counted in when determining whether a match group is single-source or multi-source. A Special source can contribute records, but it's not counted toward multi-source status. For example, some companies use a source of seed names. These are names of people who report when they receive advertising mail, so that the mailer can measure mail delivery. Appearance on the seed source is not counted toward multi-source status.

The reason for identifying the source type is to set that identity for each of the records that are members of the source. Source type plays an important role in controling priority (order) of records in break group, how the software processes matching records (the members of match groups), and how the software produces output (that is, whether it includes or excludes a record from its output).

2.16.4.7.2.1 To manually define input sources

Once you have mapped in an input field that contains the source values, you can create your sources in the Match Editor.

1. In the Match Editor, select *Transform Options* in the explorer pane on the left, click the *Add* button, and select *Input Sources*.
   The new Input Sources operation appears under Transform Options in the explorer pane. Select it to view Input Source options.
2. In the *Value field* drop-down list, choose the field that contains the input source value.
3. In the *Define sources* table, create a source name, type a source value that exists in the Value field for that source, and choose a source type.
4. Choose value from the *Default source name* option. This name will be used for any record whose source field value is blank.

Be sure to click the Apply button to save any changes you have made, before you move to another operation in the Match Editor.

2.16.4.7.2.2 To automatically define input sources

To avoid manually defining your input sources, you can choose to do it automatically by choosing the *Auto generate sources* option in the Input Sources operation.

1. In the Match Editor, select *Transform Options* in the explorer pane on the left, click the *Add* button, and select *Input Sources*.
   The new Input Sources operation appears under Transform Options in the explorer pane. Select it to view Input Source options.

2. In the *Value field* drop-down list, choose the field that contains the input source value.
3. Choose value from the *Default source name* option. This name will be used for any record whose source field value is blank.
4. Select the *Auto generate sources* option.
5. Choose a value in the *Default type* option

   The default type will be assigned to to any source that does not already have the type defined in the Type field.
6. Select a field from the drop-down list in the *Type field* option.

Auto generating sources will create a source for each unique value in the Value field. Any records that do not have a value field defined will be assigned to the default source name.

2.16.4.7.3 Source groups

The source group capability adds a higher level of source management. For example, suppose you rented several files from two brokers. You define five sources to be used in ranking the records. In addition, you would like to see your job's statistics broken down by broker as well as by file. To do this, you can define groups of sources for each broker.

Source groups primarily affect reports. However, you can also use source groups to select multi-source records based on the number of source groups in which a name occurs.

Remember that you cannot use source groups in the same way you use sources. For example, you cannot give one source group priority over another.

2.16.4.7.3.1 To create source groups

You must have input sources in an Input Source operation defined to be able to add this operation or define your source groups.

1. Select a Match transform in your data flow, and choose *Tools* > *Match Editor* .
2. In the Match Editor, select *Transform Options* in the explorer pane on the left, click the *Add* button, and select *Source Groups*.
   The new Source Groups operation appears under Input Sources operation in the explorer pane. Select it to view Source Group options.
3. Confirm that the input sources you need are in the Sources column on the right.
4. Double-click the first row in the Source Groups column on the left, and enter a name for your first source group, and press Enter.
5. Select a source in the Sources column and click the *Add* button.
6. Choose a value for the *Undefined action* option.

   This option specifies the action to take if an input source does not appear in a source group.
7. If you chose Default as the undefined action in the previous step, you must choose a value in the *Default source group* option.

   This option is populated with source groups you have already defined. If an input source is not assigned to a source group, it will be assigned to this default source group.

8. If you want, select a field in the *Source group field* option drop-down list that contains the value for your source groups.

2.16.4.8 Match preparation

2.16.4.8.1 Prepare data for matching

Data correction and standardization

Accurate matches depend on good data coming into the Match transform. For batch matching, we always recommend that you include one of the address cleansing transforms and a Data Cleanse transform in your data flow before you attempt matching.

Filter out empty records

You should filter out empty records before matching. This should help performance. Use a Case transform to route records to a different path or a Query transform to filter or block records.

Remove noise words

You can perform a search and replace on words that are meaningless to the matching process. For matching on firm data, words such as Inc., Corp., and Ltd. can be removed. You can use the search and replace function in the Query transform to accomplish this.

Remove punctuation

To maximize your matching process, map the following Data Cleanse transform output fields into your Match transform. These fields output standarized data that has been converted to uppercase, has had punctuation removed, and so on. See the Data Cleanse output field section for more information.

- Match_Family_Name
- Match_Firm
- Match_Given_Name1
- Match_Given_Name2
- Match_Maturity_Postname
- Match_Phone
- Match_Prename

Break groups

Break groups organize records into collections that are potential matches, thus reducing the number of comparisons that the Match transform must perform. Include a Break Group operation in your Match transform to improve performance.

Match standards

You may want to include variations of name or firm data in the matching process to help ensure a match. For example, a variation of “Bill” might be “William”. When making comparisons, you may want to use the original data and one or more variations. You can add anywhere from one to five variations or match standards, depending on the type of data.

For example, If the first names are compared but don't match, the variations are then compared. If the variations match, the two records still have a chance of matching rather than failing, because the original first names were not considered a match.

Custom Match Standards

You can match on custom Data Cleanse output fields and associated aliases. Map the custom output fields from Data Cleanse and the custom fields will appear in the Match Editor's Criteria Fields tab.

Related Information

Set up for match standards criteria [page 548]
Reference Guide: Output fields [page 1164]

2.16.4.8.1.1 Fields to include for matching

To take advantage of the wide range of features in the Match transform, you will need to map a number of input fields, other than the ones that you want to use as match criteria.

> Example
>
> Here are some of the other fields that you might want to include. The names of the fields are not important, as long as you remember which field contains the appropriate data.

Field contents	Contains...
Logical source	A value that specifies which logical source a record originated. This field is used in the Group Statistics operation, compare tables, and also the Associate transform.
Physical source	A value that specifies which physical source a record originated. (For example, a source object, or a group of candidate-selected records) This field is used in the Match transform options, Candidate Selection operation, and the Associate transform.
Break keys	A field that contains the break key value for creating break groups. Including a field that already contains the break key value could help improve the performance of break group creation, because it will save the Match transform from doing the parsing of multiple fields to create the break key.
Criteria fields	The fields that contain the data you want to match on.
Count flags	A Yes or No value to specify whether a logical source should be counted in a Group Statistics operation.
Record priority	A value that is used to signify a record as having priority over another when ordering records. This field is used in Group Prioritization operations.
Apply blank penalty	A Yes or No value to specify whether Match should apply a blank penalty to a record. This field is used in Group Prioritization operations.
Starting unique ID value	A starting ID value that will then increment by 1 every time a unique ID is assigned. This field is used in the Unique ID operation.

This is not a complete list. Depending on the features you want to use, you may want to include many other fields that will be used in the Match transform.

2.16.4.8.2 Control record comparisons

Controlling the number of record comparisons in the matching process is important for a couple of reasons:

- **Speed.** By controlling the actual number of comparisons, you can save processing time.
- **Match quality.** By grouping together only those records that have a potential to match, you are assured of better results in your matching process.

Controlling the number of comparisons is primarily done in the Group Forming section of the Match editor with the following operations:

- Break group: Break up your records into smaller groups of records that are more likely to match.
- Candidate selection: Select only match candidates from a database table. This is primarily used for real-time jobs.

You can also use compare tables to include or exclude records for comparison by logical source.

Related Information

Break groups [page 529]

Candidate selection [page 531]
Compare tables [page 535]

2.16.4.8.2.1 Break groups

When you create break groups, you place records into groups that are likely to match. For example, a common scenario is to create break groups based on a postcode. This ensures that records from different postcodes will never be compared, because the chances of finding a matching record with a different postcode are very small.

Break keys

You form break groups by creating a break key: a field that consists of parts of other fields or a single field, which is then used to group together records based on similar data.

Here is an example of a typical break key created by combining the five digits of the Postcode1 field and the first three characters of the Address_Primary_Name field.

Field (Start pos:length)	Data in field	Generated break key
Postcode1 (1:5)	10101	10101Mai
Address_Primary_Name (1:3)	Main	

All records that match the generated break key in this example are placed in the same break group and compared against one another.

Sorting of records in the break group

Records are sorted on the break key field.

You can add a Group Prioritization operation after the Break Groups operation to specify which records you want to be the drivers.

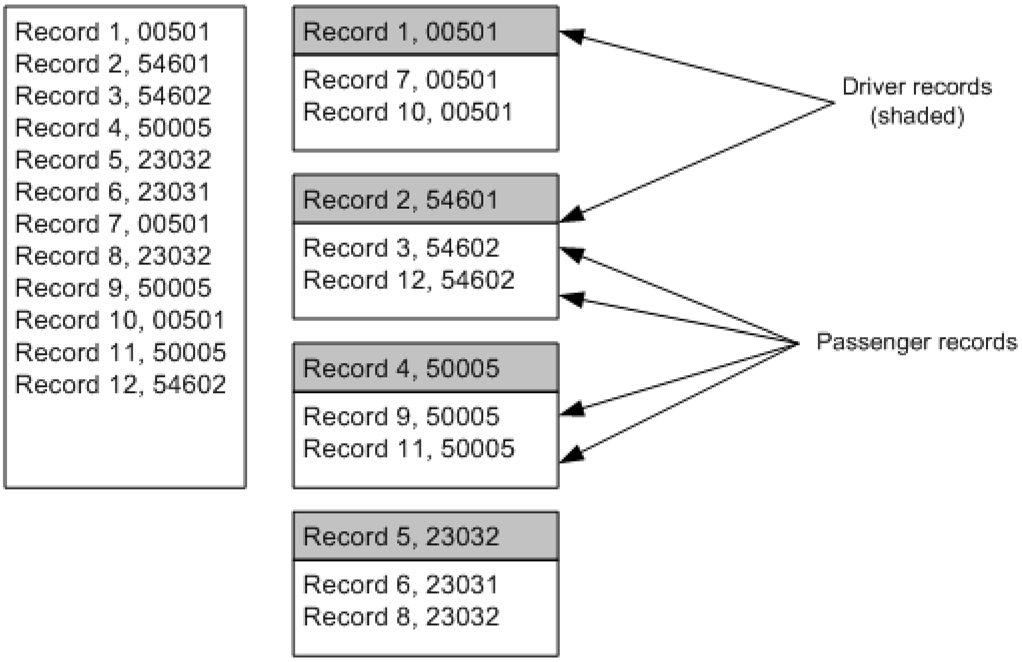
> ➜ Remember
>
> Order is important! If you are creating break groups using records from a Suppress-type source, be sure that the suppression records are the drivers in the break group.

Break group anatomy

Break groups consist of driver and passenger records. The driver record is the first record in the break group, and all other records are passengers.

The driver record is the record that drives the comparison process in matching. The driver is compared to all of the passengers first.

This example is based on a break key that uses the first three digits of the Postcode.



Phonetic break keys

You can also use the Soundex and Double Metaphone functions to create fields containing phonetic codes, which can then be used to form break groups for matching.

Related Information

Phonetic matching [page 582]
Management Console Guide: Data Quality Reports, Match Contribution report [page 2023]

2.16.4.8.2.1.1 To create break groups

We recommend that you standardize your data before you create your break keys. Data can be treated differently that is inconsistently cased, for example.

1. Add a Break Groups operation to the *Group Forming* option group.

2. in the *Break key table*, add a row by clicking the Add button.
3. Select a field in the *field* column that you want to use as a break key.
   Postcode is a common break key to use.
4. Choose the start position and length (number of characters) you want used in your break key.
   You can use negative integers to signify that you want to start at the end of the actual string length, not the specified length of the field. For example, Field(-3,3) takes the last 3 characters of the string, whether the string has length of 10 or a length of 5.
5. Add more rows and fields as necessary.
6. Order your rows by selecting a row and clicking the *Move Up* and *Move Down* buttons.
   Ordering your rows ensures that the fields are used in the right order in the break key.

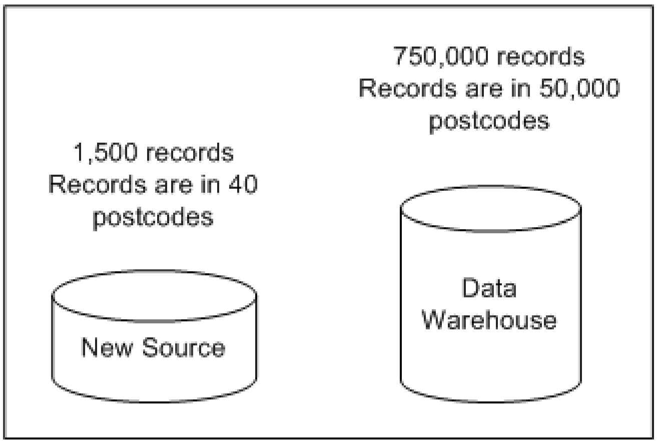
Your break key is now created.

2.16.4.8.2.2 Candidate selection

To speed processing in a match job, use the Candidate Selection operaton (Group forming option group) in the Match transform to append records from a relational database to an existing data collection before matching. When the records are appended, they are not logically grouped in any way. They are simply appended to the end of the data collection on a record-by-record basis until the collection reaches the specified size.

For example, suppose you have a new source of records that you want to compare against your data warehouse in a batch job. From this warehouse, you can select records that match the break keys of the new source. This helps narrow down the number of comparisons the Match transform has to make.

For example, here is a simplified illustration: Suppose your job is comparing a new source database—a smaller, regional file—with a large, national database that includes 15 records in each of 43,000 or so postcodes. Further assume that you want to form break groups based only on the postcode.



Notes	Regional	National	Total
Without candidate selection, the Match transform reads all of the records of both databases.	1,500	750,000	751,500

Notes	Regional	National	Total
With candidate selection, only those records that would be included in a break group are read.	1,500	About 600 (40 x 15)	2,100

2.16.4.8.2.2.1 Datastores and candidate selection

To use candidate selection, you must connect to a valid datastore. You can connect to any SQL-based or persistent cache datastore. There are advantages for using one over the other, depending on whether your secondary source is static (it isn't updated often) or dynamic (the source is updated often).

Persistent cache datastores

Persistent cache is like any other datastore from which you can load your candidate set. If the secondary source from which you do candidate selection is fairly static (that is, it will not change often), then you might want consider building a persistent cache, rather than using your secondary source directly, to use as your secondary table. You may improve performance.

You may also encounter performance gains by using a flat file (a more easily searchable format than a RDBMS) for your persistent cache. If the secondary source is not an RDBMS, such as a flat file, you cannot use it as a "datastore". In this case, you can create a persistent cache out of that flat file source and then use that for candidate selection.

i Note

A persistent cache used in candidate selection must be created by a data flow in double-byte mode. To do this, you will need to change the locale setting in the Data Services Locale Selector (set the code page to utf-8). Run the job to generate the persistent cache, and then you can change the code page back to its original setting if you want.

Cache size

Performance gains using persistent cache also depend on the size of the secondary source data. As the size of the data loaded in the persistent cache increases, the performance gains may decrease. Also note that if the original secondary source table is properly indexed and optimized for speed then there may be no benefit in creating a persistent cache (or even pre-load cache) out of it.

Related Information

Persistent cache datastores [page 229]

2.16.4.8.2.2.2 Auto-generation vs. custom SQL

There are cases where the Match transform can generate SQL for you, and there are times where you must create your own SQL. This is determined by the options you and how your secondary table (the table you are selecting match candidates from) is set up.

Use this table to help you determine whether you can use auto-generated SQL or if you must create your own.

> Note
>
> In the following scenarios, “input data” refers to break key fields coming from a transform upstream from the Match transform (such as a Query transform) or a break key fields coming from the Break Group operation within the Match transform itself.

Scenario	Auto-generate or Custom?
You have a single break key field in your input data, and you have the same field in your secondary table.	Auto-generate
You have multiple break key fields in your input data, and you have the same fields in your secondary table.	Auto-generate
You have multiple break key fields in your input data, and you have one break key field in your secondary table.	Auto-generate
You have a single break key field in your input data, and you have multiple break key fields in your secondary table.	Custom
You have multiple break key fields in your input data, but you have a different format or number of fields in your secondary table.	Custom
You want to select from multiple input sources.	Custom

2.16.4.8.2.2.3 Break keys and candidate selection

We recommend that you create a break key column in your secondary table (the table that contains the records you want to compare with the input data in your data flow) that matches the break key you create your break groups with in the Match transform. This makes setup of the Candidate Selection operation much easier. Also, each of these columns should be indexed.

We also recommend that you create and populate the database you are selecting from with a single break key field, rather than pulling substrings from database fields to create your break key. This can help improve the performance of candidate selection.

> Note
>
> Records extracted by candidate selection are appended to the end of an existing break group (if you are using break groups). So, if you do not reorder the records using a Group Prioritization operation after the Candidate Selection operation, records from the original source will always be the driver records in the break groups. If you are using candidate selection on a Suppress source, you will need to reorder the records so that the records from the Suppress source are the drivers.

2.16.4.8.2.2.4 To set up candidate selection

If you are using Candidate selection for a real-time job, be sure to deselect the *Split records into break groups* option in the Break Group operation of the Match transform.

To speed processing in a real-time match job, use the Candidate Selection operaton (Group forming option group) in the Match transform to append records from a relational database to an existing data collection before matching. When the records are appended, they are not logically grouped in any way. They are simply appended to the end of the data collection on a record-by-record basis until the collection reaches the specified size.

1. In the Candidate Selection operation, select a valid datastore from the *Datastore* drop-down list.
2. In the *Cache type* drop-down list, choose from the following values:

Option	Description
No_Cache	Captures data at a point in time. The data doesn't change until the job restarts.
Pre-load Cache	Use this option for static data.

3. Depending on how your input data and secondary table are structured, do one of the following:
   - Select *Auto-generate SQL*. Then select the *Use break column from database* option, if you have one, and choose a column from the *Break key field* drop-down list.

     > Note
     >
     > If you choose the *Auto-generate SQL* option, we recommend that you have a break key column in your secondary table and select the *Use break column from database* option. If you don't, the SQL that is created could be incorrect.

   - Select *Create custom SQL*, and either click the *Launch SQL Editor* button or type your *SQL* in the SQL edit box.
4. If you want to track your records from the input source, select *Use constant source value*.
5. Enter a value that represents your source in the *Physical source value* option, and then choose a field that holds this value in the *Physical source field* drop-down list.
6. In the *Column mapping* table, add as many rows as you want. Each row is a field that will be added to the collection.
   a) Choose a field in the *Mapped name* column.
   b) Choose a column from your secondary table (or from a custom query) in the *Column name* option that contains the same type of data as specified in the *Mapped name* column.

   If you have already defined your break keys in the Break Group option group, the fields used to create the break key are posted here, with the Break Group column set to YES.

2.16.4.8.2.2.5 Writing custom SQL

Use placeholders

To avoid complicated SQL statements, you should use placeholders (which are replaced with real input data) in your WHERE clause.

For example, let's say the customer database contains a field called MatchKey, and the record that goes through the cleansing process gets a field generated called MATCH_KEY. This field has a placeholder of [MATCHKEY]. The records that are selected from the customer database and appended to the existing data collection are those that contain the same value in MatchKey as in the transaction's MATCH_KEY. For this example, let's say the actual value is a 10-digit phone number.

The following is an example of what your SQL would look like with an actual phone number instead of the [MATCHKEY] placeholder.

```
SELECT ContactGivenName1, ContactGivenName2, ContactFamilyName, Address1, Address2,
City, Region, Postcode, Country, AddrStreet, AddrStreetNumber, AddrUnitNumber

FROM TblCustomer

WHERE MatchKey = '123-555-9876';
```

Caution

You must make sure that the SQL statement is optimized for best performance and will generate valid results. The Candidate Selection operation does not do this for you.

Replace placeholder with actual values

After testing the SQL with actual values, you must replace the actual values with placeholders ([MATCHKEY], for example).

Your SQL should now look similar to the following.

```
SELECT ContactGivenName1, ContactGivenName2, ContactFamilyName, Address1, Address2,
City, Region, Postcode, Country, AddrStreet, AddrStreetNumber, AddrUnitNumber

FROM TblCustomer

WHERE MatchKey = [MATCHKEY];
```

Note

Placeholders cannot be used for list values, for example in an IN clause:

```
WHERE status IN ([status])
```

If [status] is a list of values, this SQL statement will fail.

2.16.4.8.2.3 Compare tables

Compare tables are sets of rules that define which records to compare, sort of an additional way to create break groups. You use your logical source values to determine which records are compared or are not compared.

By using compare tables, you can compare records within sources, or you can compare records across sources, or a combination of both.

2.16.4.8.2.3.1 To set up a compare table

Be sure to include a field that contains a logical source value before you add a Compare table operation to the Match transform (in the Match level option group).

Here is an example of how to set up your compare table. Suppose you have two IDs (A and B), and you only want to compare across sources, not within the sources.

1. If no Compare Table is present in the Matching section, right-click *Matching* > **`<Level Name>`**, and select *Add* > *Compare* .
2. Set the *Default action* option to *No_Match*, and type **`None`** in the *Default logical source value* option.
   This tells the Match transform to not compare everything, but follow the comparison rules set by the table entries.

   > Note
   >
   > Use care when choosing logical source names. Typing “None” in the *Default logical source value* option will not work if you have a source ID called “None.”

3. In the *Compare actions* table, add a row, and then set the *Driver value* to *A*, and set the *Passenger value* to *B*.
4. Set *Action* to *Compare*.

   > Note
   >
   > Account for all logical source values. The example values entered above assumes that A will always be the driver ID. If you expect that a driver record has a value other than A, set up a table entry to account for that value and the passenger ID value. Remember that the driver record is the first record read in a collection.
   >
   > If you leave the *Driver value* or *Passenger value* options blank in the compare table, then it will mean that you want to compare all sources. So a Driver value of A and a blank passenger record with an action of compare will make a record from A compare against all other passenger records.

Sometimes data in collections can be ordered (or not ordered, as the case may be) differently than your compare table is expecting. This can cause the matching process to miss duplicate records.

In the example, the way you set up your Compare action table row means that you are expecting that the driver record should have a driver value of A, but if the driver record comes in with a value of B, and the passenger comes in with a value of A, it won't be compared.

To account for situations where a driver record might have a value of B and the passenger a value of A, for example, include another row in the table that does the opposite. This will make sure that any record with a value of A or B is compared, no matter which is the Driver or Passenger.

> Note
>
> In general, if you use a suppress source, you should compare within the other sources.This ensures that all of the matches of those sources are suppressed when any are found to duplicate a record on the suppress source, regardless of which record is the driver record.

2.16.4.8.3 Order and prioritize records

You may have data sources, such as your own data warehouse, that you might trust more than records from another source, such as a rented source, for example. You may also prefer newer records over older records, or more complete records over those with blank fields. Whatever your preference, the way to express this preference in the matching process is using priorities.

There are other times where you might want to ensure that your records move to a given operation, such as matching or best record, for example, in a particular order. For example, you might want your match groups to be ordered so that the first record in is the newest record of the group. In this case, you would want to order your records based on a date field.

Whatever the reason, there are a two ways to order your records, either before or after the comparison process:

- Sorting records in break groups or match groups using a value in a field.
- Using penalty scores. These can be defined per field, per record, or based on input source membership.

Match editor

You can define your priorities and order your records in the Group Prioritization operation, available in Group Forming and in the Post-match processing operations of each match level in the Match editor.

Types of priorities

There are a couple of different types of priorities to consider:

Priority	Brief description
Record priority	Prefers records from one input source over another.
Blank penalty	Assigns a lower priority to records in which a particular field is blank.

Pre-match ordering

When you create break groups, you can set up your *Group Forming* > *Group Prioritization* operation to order (or sort) on a field, besides ordering on the break key. This will ensure that the highest priority record is the first record (driver) in the break group.

You will also want to have Suppress-type input sources to be the driver records in a break group.

Post-match ordering

After the Match transform has created all of the match groups, and if order is important, you can use a Group Prioritization operation before a Group Statistics, Best Record, and Unique ID operations to ensure that the master record is the first in the match group.

> Tip
>
> If you are not using a blank penalty, order may not be as important to you, and you may not want to include a Group Prioritization operation before your post-match operations. However, you may get better performance out of a Best Record operation by prioritizing records and then setting the *Post only once per destination* option to Yes.

Blank penalty

Given two records, you may prefer to keep the record that contains the most complete data. You can use blank penalty to penalize records that contain blank fields.

Incorporating a blank penalty is appropriate if you feel that a blank field shouldn't disqualify one record from matching another, and you want to keep the more complete record. For example, suppose you are willing to accept a record as a match even if the Prename, Given_Name1, Given_Name2, Primary_Postfix and/or Secondary Number is blank. Even though you accept these records into your match groups, you can assign them a lower priority for each blank field.

2.16.4.8.3.1 To order records by sorting on a field

Be sure you have mapped the input fields into the Match transform that you want to order on, or they won't show up in the field drop-down list.

Use this method of ordering your records if you do not consider completeness of data important.

1. Enter a Prioritization name, and select the Priority Order tab.
2. In the *Priority fields* table, choose a field from the drop-down list in the *Input Fields* column.
3. In the *Field Order* column, choose *Ascending* or *Descending* to specify the type of ordering.
   For example, if you are comparing a Normal source to a Suppress source and you are using a source ID field to order your records, you will want to ensure that records from the Suppress source are first in the break group.
4. Repeat step 2 for each row you added.
5. Order your rows in the *Priority fields* table by using the *Move Up* and *Move Down* buttons.

The first row will be the primary order, and the rest will be secondary orders.

2.16.4.8.3.2 Penalty scoring system

The blank penalty is a penalty-scoring system. For each blank field, you can assess a penalty of any non-negative integer.

You can assess the same penalty for each blank field, or assess a higher penalty for fields you consider more important. For example, if you were targeting a mailing to college students, who primarily live in apartments or dormitories, you might assess a higher penalty for a blank Given_Name1 or apartment number.

Field	Blank penalty
Prename	5
Given_Name1	20
Given_Name2	5
Primary Postfix	5
Secondary Number	20

As a result, the records below would be ranked in the order shown (assume they are from the same source, so record priority is not a factor). Even though the first record has blank prename, Given_Name2, and street postfix fields, we want it as the master record because it does contain the data we consider more important: Given_Name1 and Secondary Number.

Prename (5)	Given Name1 (20)	Given Name2 (5)	Family Name	Prim Range	Prim Name	Prim Postfix (5)	Sec Number (20)	Blank-field penalty
	Maria		Ramirez	100	Main		6	5 + 5 + 5 = 15
Ms.	Maria	A	Ramirez	100	Main	St		20
Ms.			Ramirez	100	Main	St	6	20 + 5 = 25

2.16.4.8.3.3 Blank penalty interacts with record priority

The record priority and blank penalty scores are added together and considered as one score.

For example, suppose you want records from your house database to have high priority, but you also want records with blank fields to have low priority. Is source membership more important, even if some fields are blank? Or is it more important to have as complete a record as possible, even if it is not from the house database?

Most want their house records to have priority, and would not want blank fields to override that priority. To make this happen, set a high penalty for membership in a rented source, and lower penalties for blank fields:

Source	Record priority (penalty points)	Field	Blank penalty
House Source	100	Given Name1	20
Rented Source A	200	Given_Name2	5
Rented Source B	300	Primary Postfix	5
Rented Source C	400	Secondary Number	20

With this scoring system, a record from the house source always receives priority over a record from a rented source, even if the house record has blank fields. For example, suppose the records below were in the same match group.

Even though the house record contains five blank fields, it receives only 155 penalty points (100 + 5 + 20 + 5 + 5 + 20), while the record from source A receives 200 penalty points. The house record, therefore, has the lower penalty and the higher priority.

Source	Given Name1	Given Name2	Family	Prim Range	Prim Name	Sec Num	Post-code	Rec pri-ority	Blank Penalty	Total
House			Smith	100	Bren		55343	100	55	155
Source A	Rita	A	Smith	100	Bren	12A	55343	200	0	200
Source B	Rita		Smith	100	Bren	12	55343	300	10	310

You can manipulate the scores to set priority exactly as you'd like. In the example above, suppose you prefer a rented record containing first-name data over a house record without first-name data. You could set the first-name blank penalty score to 500 so that a blank first-name field would weigh more heavily than any source membership.

2.16.4.8.3.4 To define priority and penalty using field values

Be sure to map in any input fields that carry priority or blank penalty values.

This task tells Match which fields hold your record priority and blank penalty values for your records, and whether to apply these per record.

1. Add a Group Prioritization operation to the Group Forming or Post Match Processing section in the Match Editor.
2. Enter a Prioritization name (if necessary) and select the *Record Completeness* tab.
3. Select the *Order records based on completeness of data* option.
4. Select the *Define priority and penalty fields* option.
    - *Define only field penalties:* This option allows you to select a default record priority and blank penalties per field to generate your priority score.
    - *Define priority and penalty based on input source:* This allows you to define priority and blank penalty based on membership in an input source.

5. Choose a field that contains the record priority value from the *Record priority field* option.
6. In the *Apply blank penalty field* option, choose a field that contains the Y or N indicator for whether to apply a blank penalty to a record.
7. In the *Default record priority* option, enter a default record priority to use if a record priority field is blank or if you do not specify a record priority field.
8. Choose a *Default apply blank penalty* value (Yes or No). This determines whether the Match transform will apply blank penalty to a record if you didn't choose an apply blank penalty field or if the field is blank for a particular record.
9. In the *Blank penalty score* table, choose a field from the *Input Field* column to which you want to assign blank penalty values.
10. In the *Blank Penalty* column, type a blank penalty value to attribute to any record containing a blank in the field you indicated in *Input Field* column.

2.16.4.8.3.5 To define penalty values by field

This task lets you define your default priority score for every record and blank penalties per field to generate your penalty score.

1. Add a Group Prioritization operation to the Group Forming or Post Match Processing section in the Match Editor.
2. Enter a Prioritization name (if necessary) and select the *Record Completeness* tab.
3. Select the *Order records based on completeness of data* option.
4. Select the *Define only field penalties* option.
5. In the *Default record priority* option, enter a default record priority that will be used in the penalty score for every record.
6. Choose a *Default apply blank penalty* value (Yes or No). This determines whether the Match transform will apply blank penalty to a record if you didn't choose an apply blank penalty field or if the field is blank for a particular record.
7. In the *Blank penalty score* table, choose a field from the *Input Field* column to which you want to assign blank penalty values.
8. In the *Blank Penalty* column, type a blank penalty value to attribute to any record containing a blank in the field you indicated in *Input Field* column.

2.16.4.8.4 Prioritize records based on source membership

However you prefer to prioritize your sources (by sorting a break group or by using penalty scores), you will want to ensure that your suppress-type source records are the drivers in the break group and comparison process.

For example, suppose you are a charitable foundation mailing a solicitation to your current donors and to names from two rented sources. If a name appears on your house source and a rented source, you prefer to use the name from your house source.

For one of the rented sources, Source B, suppose also that you can negotiate a rebate for any records you do not use. You want to use as few records as possible from Source B so that you can get the largest possible rebate.

Therefore, you want records from Source B to have the lowest preference, or priority, from among the three sources.

Source	Priority
House source	Highest
Rented source A	Medium
Rented source B	Lowest

Suppress-type sources and record completeness

In cases where you want to use penalty scores, you will want your Suppress-type sources to have a low priority score. This makes it likely that normal records that match a suppress record will be subordinate matches in a match group, and will therefore be suppressed, as well. Within each match group, any record with a lower priority than a suppression source record is considered a suppress match.

For example, suppose you are running your files against the DMA Mail Preference File (a list of people who do not want to receive advertising mailings). You would identify the DMA source as a suppression source and assign a priority of zero.

Source	Priority
DMA Suppression source	0
House source	100
Rented source A	200
Rentd source B	300

Suppose Match found four matching records among the input records.

Matching record (name fields only)				Source	Priority
	Maria		Ramirez	House	100
Ms.			Ramirez	Source B	300
Ms.	Maria	A	Ramirez	Source A	200
Ms.	Maria	A	Ramirez	DMA	0

The following match group would be established. Based on their priority, Match would rank the records as shown. As a result, the record from the suppression file (the DMA source) would be the master record, and the others would be subordinate suppress matches, and thus suppressed, as well.

Source	Priority
DMA	0 (Master record)
House	100
Source A	200
Source B	300

2.16.4.8.4.1 To define penalties based on source membership

In this task, you can attribute priority scores and blank penalties to an input source, and thus apply these scores to any record belonging to that source. Just be sure you have your input sources defined before you attempt to complete this task.

1. Add a Group Prioritization operation to the Group Forming or Post Match Processing section in the Match Editor.
2. Enter a Prioritization name (if necessary) and select the *Record Completeness* tab.
3. Select the *Order records based on completeness of data* option.
4. Select the *Define priority and penalty based on input source* option.
5. In the *Source Attributes* table, select a source from the drop-down list.
6. Type a value in the *Priority* column to assign a record priority to that source.

   Remember that the lower the score, the higher the priority. For example, you would want to assign a very low score (such as 0) to a suppress-type source.
7. In the *Apply Blank Penalty* column, choose a Yes or No value to determine whether to use blank penalty on records from that source.
8. In the *Default record priority* option, enter a default record priority that will be used in the penalty score for every record that is not a member of a source.
9. Choose a *Default apply blank penalty* value (Yes or No). This determines whether to apply blank penalties to a record that is not a member of a source.
10. In the *Blank penalty score* table, choose a field from the *Input Field* column to which you want to assign blank penalty values.
11. In the *Blank Penalty* column, type a blank penalty value to attribute to any record containing a blank in the field you indicated in *Input Field* column.

2.16.4.8.5 Data Salvage

Data salvaging temporarily copies data from a passenger record to the driver record after comparing the two records. The data that's copied is data that is found in the passenger record but is missing or incomplete in the driver record. Data salvaging prevents blank matching or initials matching from matching records that you may not want to match.

For example, we have the following match group. If you did not enable data salvaging, the records in the first table would all belong to the same match group because the driver record, which contains a blank Name field, matches both of the other records.

Record	Name	Address	Postcode
1 (driver)		123 Main St.	54601
2	John Smith	123 Main St.	54601
3	Jack Hill	123 Main St.	54601

If you enabled data salvaging, the software would temporarily copy John Smith from the second record into the driver record. The result: Record #1 matches Record #2, but Record #1 does not match Record #3 (because John Smith doesn't match Jack Hill).

Record	Name	Address	Postcode
1 (driver)	John Smith (copied from record below)	123 Main St.	54601
2	John Smith	123 Main St.	54601
3	Jack Hill	123 Main St.	54601

The following example shows how this is used for a suppression source. Assume that the suppression source is a list of no-pandering addresses. In that case, you would set the suppression source to have the highest priority, and you would not enable data salvaging. That way, the software suppresses all records that match the suppression source records.

For example, a suppress record of 123 Main St would match 123 Main St #2 and 123 Main St Apt C; both of these would be suppressed.

2.16.4.8.5.1 Data salvaging and initials

When a driver record's name field contains an initial, instead of a full name, the software may temporarily borrow the full name if it finds one in the corresponding field of a matching record. This is one form of data salvaging.

For illustration, assume that the following three records represent potentially matching records (for example, the software has grouped these as members of a break group, based on address and ZIP Code data).

Note

Initials salvaging only occurs with the given name and family name fields.

Record	First name	Last name	Address	Notes
357	J	L	123 Main	Driver
391	Juanita	Lopez	123 Main	
839	Joanne	London	123 Main	Lowest ranking record

The first match comparison will be between the driver record (357) and the next highest ranking record (391). These two records will be called a match. Juanita and Lopez are temporarily copied to the name fields of record# 357.

The next comparison will be between record 357 and the next lower ranking record (839). With data salvaging, the driver record's name data is now Juanita Lopez (as "borrowed" from the first comparison). Therefore, record 839 will probably be considered not-to match record 357.

By retaining more information for the driver record, data salvaging helps improve the quality of your matching results.

Initials and suppress-type records

However, if the driver record is a suppress-type record, you may prefer to turn off data salvaging, to retain your best chance of identifying all the records that match the initialized suppression data. For example, if you want to suppress names with the initials JL (as in the case above, you would want to find all matches to JL regardless of the order in which the records are encountered in the break group.

If you have turned off data salvaging for the records of this suppression source, here is what happens during those same two match comparisons:

Record	First name	Last name	Address	Notes
357	J	L	123 Main	Driver
391	Juanita	Lopez	123 Main	
839	Joanne	London	123 Main	Lowest ranking record

The first match comparison will be between the driver record (357) and the next- highest ranking record (391). These two records will be called a match, since the driver record’s JL and Juanita Lopez will be called a match.

The next comparison will be between the driver record (357) and the next lower ranking record (839). This time these two records will also be called a match, since the driver record’s JL will match Joanne London.

Since both records 391 and 839 matched the suppress-type driver record, they are both designated as suppress matches, and, therefore, neither will be included in your output.

2.16.4.8.5.2 To control data salvaging using a field

You can use a field to control whether data salvage is enabled. If the field's value is Y for a record, data salvaging is enabled. Be sure to map the field into the Match transform that you want to use beforehand.

1. Open the Match Editor for a Match transform.
2. In the Transform Options window, click the *Data Salvage* tab.
3. Select the Enable data salvage option, and choose a default value for those records.
   The default value will be used in the cases where the field you choose is not populated for a particular record.
4. Select the *Specify data salvage by field* option, and choose a field from the drop-down menu.

2.16.4.8.5.3 To control data salvaging by source

You can use membership in an input source to control whether data salvage is enabled or disabled for a particular record. Be sure to create your input sources beforehand.

1. Open the Match Editor for a Match transform.
2. In the Transform Options window, click the *Data Salvage* tab.
3. Select the *Enable data salvage* option, and choose a default value for those records.
   The default value will be used if a record's input source is not specified in the following steps.
4. Select the *Specify data salvage by source* option.
5. In the table, choose a Source and then a Perform Data Salvage value for each source you want to use.

2.16.4.9 Match criteria

2.16.4.9.1 Overview of match criteria

Use match criteria in each match level to determine the threshold scores for matching and to define how to treat various types of data, such as numeric, blank, name data, and so on (your business rules).

You can do all of this in the Criteria option group of the Match Editor.

Match criteria

To the Match transform, match criteria represent the fields you want to compare. For example, if you wanted to match on the first ten characters of a given name and the first fifteen characters of the family name, you must create two criteria that specify these requirements.

Criteria provide a way to let the Match transform know what kind of data is in the input field and, therefore, what types of operations to perform on that data.

Pre-defined vs. custom criteria

There are two types of criteria:

- Pre-defined criteria are available for fields that are typically used for matching, such as name, address, and other data. By assigning a criteria to a field, the Match transform is able to identify what type of data is in the field, and allow it to perform internal operations to optimize the data for matching, without altering the actual input data.
- Data Cleanse custom (user-defined, non party-data) output fields are available as pre-defined criteria. Map the custom output fields from Data Cleanse and the custom fields appear in the Match Editor's Criteria Fields tab.
  Any other types of data (such as part numbers or other proprietary data), for which a pre-defined criteria does not exist, should be designated as a custom criteria. Certain functions can be performed on custom keys, such as abbreviation, substring, numeric matching, but the Match transform cannot perform some cross-field comparisons such as some name matching functions.

Match criteria pre-comparison options

The majority of your data standardization should take place in the address cleansing and Data Cleanse transforms. However, the Match transform can perform some preprocessing per criteria (and for matching purposes only; your actual data is not affected) to provide more accurate matches. The options to control this standardization are located in the Options and Multi Field Comparisons tabs of the Match editor. They include:

- Convert diacritical characters
- Convert text to numbers

- Convert to uppercase
- Remove punctuation
- Locale

For more information about these options, see the Match transform section of the *Reference Guide*.

2.16.4.9.1.1 To add and order a match criteria

You can add as many criteria as you want to each match level in your Match transform.

1. Select the appropriate match level or Match Criteria option group in the Option Explorer of the Match Editor, and right-click.
2. Choose *Criteria*.
3. Enter a name for your criteria in the *Criteria name* box.

   You can keep the default name for pre-defined criteria, but you should enter a meaningful criteria name if you chose a Custom criteria.
4. On the *Criteria Fields* tab, in the *Available criteria* list, choose the criteria that best represents the data that you want to match on. If you don't find what you are looking for, choose the Custom criteria.
5. In the *Criteria field mapping* table, choose an input field mapped name that contains the data you want to match on for this criteria.
6. Click the *Options* tab.
7. Configure the *Pre-comparison options* and *Comparison rules*.

   Be sure to set the Match score and No match score, because these are required.
8. If you want to enable multiple field (cross-field) comparison, click the *Multiple Fields Comparisons* tab, and select the *Compare multiple fields* option.
   a) Choose the type of multiple field comparison to perform:
      - *All selected fields in other records*: Compare each field to all fields selected in the table in all records.
      - *The same field in other records*: Compare each field only to the same field in all records.

   b) In the *Additional fields to compare* table, choose input fields that contain the data you want to include in the multiple field comparison for this criteria.

   > ➜ Tip
   >
   > You can use custom match criteria field names for multiple field comparison by typing in the *Custom name* column.

   > i Note
   >
   > If you enable multiple field comparison, any appropriate match standard fields are removed from the *Criteria field mapping* table on the *Criteria Fields* tab . If you want to include them in the match process, add them in the *Additional fields to compare* table.
9. Configure the *Pre-comparison* options for multiple field comparison.
10. To order your criteria in the Options Explorer of the Match Editor (or the Match Table), select a criteria and click the *Move Up* or *Move Down* buttons as necessary.

2.16.4.9.1.2 Set up for match standards criteria

Be sure you have created and mapped in match standard fields from the Data Cleanse transform.

1. In the Match transform, add a Person1_Given_Name1 match criteria by selecting it from the Person category. The editor automatically adds that criteria and its associated match standard criteria to the *Criteria field mapping* table.
2. Choose a field in the *Input field mapped name* column that best represents the data to be compared.

For Given_Name2 data (middle names), complete the above procedure, but use the following criteria:

- Person1_Given_Name2
- Person1_Given_Name2_Match_Std1
- Person1_Given_Name2_Match_Std2
- Person1_Given_Name2_Match_Std3

You can also use match standards for prename, maturity postname, honorary postname, firm, and firm location fields.

2.16.4.9.2 Matching methods

There are a number of ways to set up and order your criteria to get the matching results you want. Each of these ways have advantages and disadvantages, so consider them carefully.

Match method	Description
Rule-based	Allows you to control which criteria determines a match. This method is easy to set up.
Weighted-scoring	Allows you to assign importance, or weight, to any criteria. However, weighted-scoring evaluates every rule before determining a match, which might cause an increase in processing time.
Combination method	Same relative advantages and disadvantages as the other two methods.

2.16.4.9.2.1 Similarity score

The similarity score is the percentage that your data is alike. This score is calculated internally by the application when records are compared. Whether the application considers the records a match depends on the Match and No match scores you define in the Criteria option group (as well as other factors, but for now let's focus on these scores).

Example

This is an example of how similarity scores are determined. Here are some things to note:

- The comparison table below is intended to serve as an example. This is not how the matching process works in the weighted scoring method, for example.

- Only the first comparison is considered a match, because the similarity score met or exceeded the match score. The last comparison is considered a no-match because the similarity score was less than the no-match score.
- When a single criteria cannot determine a match, as in the case of the second comparison in the table below, the process moves to the next criteria, if possible.

Comparison	No match	Match	Similarity score	Matching?
Smith > Smith	72	95	100%	Yes
Smith > Smitt	72	95	80%	Depends on other criteria
Smith > Smythe	72	95	72%	No
Smith > Jones	72	95	20%	No

2.16.4.9.2.2 Rule-based method

With rule-based matching, you rely only on your match and no-match scores to determine matches within a criteria.

Example

This example shows how to set up this method in the Match transform.

Criteria	Record A	Record B	No match	Match	Similarity score
Given Name1	Mary	Mary	82	101	100
Family Name	Smith	Smitt	74	101	80
E-mail	msmith@sap.com	mary.smith@sap.com	79	80	91

By entering a value of 101 in the match score for every criteria except the last, the Given Name1 and Family Name criteria never determine a match, although they can determine a no match.

By setting the *Match score* and *No match score* options for the E-mail criteria with no gap, any comparison that reaches the last criteria must either be a match or a no match.

A match score of 101 ensures that the criteria does not cause the records to be a match, because two fields cannot be more than 100 percent alike.

Remember

Order is important! For performance reasons, you should have the criteria that is most likely to make the match or no-match decisions first in your order of criteria. This can help reduce the number of criteria comparisons.

2.16.4.9.2.3 Weighted-scoring method

In a rule-based matching method, the application gives all of the criteria the same amount of importance (or weight). That is, if any criteria fails to meet the specified match score, the application determines that the records do not match.

When you use the weighted scoring method, you are relying on the total contribution score for determining matches, as opposed to using match and no-match scores on their own.

Contribution values

Contribution values are your way of assigning weight to individual criteria. The higher the value, the more weight that criteria carries in determining matches. In general, criteria that might carry more weight than others include account numbers, Social Security numbers, customer numbers, Postcode1, and addresses.

> Note
>
> All contribution values for all criteria that have them must total 100. You do not need to have a contribution value for all of your criteria.

You can define a criteria's contribution value in the Contribution to weighted score option in the Criteria option group.

Contribution and total contribution score

The Match transform generates the contribution score for each criteria by multiplying the contribution value you assign with the similarity score (the percentage alike). These individual contribution scores are then added to get the total contribution score.

Weighted match score

In the weighted scoring method, matches are determined only by comparing the total contribution score with the weighted match score. If the total contribution score is equal to or greater than the weighted match score, the records are considered a match. If the total weighted score is less than the weighted match score, the records are considered a no-match.

You can set the weighted match score in the *Weighted match score* option of the Level option group.

> Example
>
> The following table is an example of how to set up weighted scoring. Notice the various types of scores that we have discussed. Also notice the following:

- When setting up weighted scoring, the *No match score* option must be set to -1, and the *Match score* option must be set to 101. These values ensure that neither a match nor a no-match can be found by using these scores.
- We have assigned a contribution value to the E-mail criteria that gives it the most importance.

Criteria	Record A	Record B	No match	Match	Similarity score	Contribution value	Contribution score (similarity X contribution value)
First Name	Mary	Mary	-1	101	100	25	25
Last Name	Smith	Smitt	-1	101	80	25	20
E-mail	ms@ sap.com	msmith@ sap.com	-1	101	84	50	42
							Total contribution score: 87

If the weighted match score is 87, then any comparison whose total contribution score is 87 or greater is considered a match. In this example, the comparison is a match because the total contribution score is 87.

2.16.4.9.2.4 Combination method

This method combines the rule-based and weighted scoring methods of matching.

Criteria	Record A	Record B	No match	Match	Sim score	Contribution value	Contribution score (actual similarity X contribution value)	
First Name	Mary	Mary	59	101	100	25		25
Last Name	Smith	Hope	**59**	101	**22**	N/A **(No Match)**		N/A
E-mail	ms@ sap.com	msmith@ sap.com	49	101	N/A	N/A		N/A
							Total contribution score	N/A

2.16.4.9.3 Matching business rules

An important part of the matching process is determining how you want to handle various forms of and differences in your data. For example, if every field in a record matched another record's fields, except that one field was blank and the other record's field was not, would you want these records to be considered matches?

Figuring out what you want to do in these situations is part of defining your business rules. Match criteria are where you define most of your business rules, while some name-based options are set in the Match Level option group.

2.16.4.9.3.1 Matching on strings, abbreviations, and initials

Initials and acronyms

Use the *Initials adjustment score* option to allow matching initials to whole words. For example, "International Health Providers" can be matched to "IHP".

Abbreviations

Use the *Abbreviation adjustment score* option to allow matching whole words to abbreviations. For example, "International Health Providers" can be matched to "Intl Health Providers".

String data

Use the *Substring adjustment score* option to allow matching longer strings to shorter strings. For example, the string "Mayfield Painting and Sand Blasting" can match "Mayfield painting".

2.16.4.9.3.2 Extended abbrevation matching

Extended abbreviation matching offers functionality that handles situations not covered by the *Initials adjustment score*, *Substring adjustment score*, and *Abbreviation adjustment score* options. For example, you might encounter the following situations:

- Suppose you have localities in your data such as La Crosse and New York. However, you also have these same localities listed as LaCrosse and NewYork (without spaces). Under normal matching, you cannot designate these (La Crosse/LaCrosse and New York/NewYork) as matching 100%; the spaces prevent this. (These would normally be 94 and 93 percent matching.)
- Suppose you have Metropolitan Life and MetLife (an abbreviation and combination of Metropolitan Life) in your data. The *Abbreviation adjustment score* option cannot detect the combination of the two words.

If you are concerned about either of these cases in your data, you should use the *Ext abbreviation adjustment score* option.

How the adjustment score works

The score you set in the *Ext abbreviation adjustment score* option tunes your similarity score to consider these types of abbreviations and combinations in your data.

The adjustment score adds a penalty for the non-matched part of the words. The higher the number, the lower the penalty. A score of 100 means no penalty and score of 0 means maximum penalty.

Example

String 1	String 2	Sim score when Adj score is 0	Sim score when Adj score is 50	Sim score when Adj score is 100	Notes
MetLife	Metropolitan Life	58	79	100	
MetLife	Met Life	93	96	100	
MetLife	MetropolitanLife	60	60	60	This score is due to string comparison. Extended Abbreviation scoring was not needed or used because both strings being compared are each one word.

2.16.4.9.3.3 Name matching

Part of creating your business rules is to define how you want names handled in the matching process. The Match transform gives you many ways to ensure that variations on names or multiple names, for example, are taken into consideration.

Note

Unlike other business rules, these options are set up in the match level option group, because they affect all appropriate name-based match criteria.

Two names; two persons

With the *Number of names that must match* option, you can control how matching is performed on match keys with more than one name (for example, comparing "John and Mary Smith" to "Dave and Mary Smith"). Choose whether only one name needs to match for the records to be identified as a match, or whether the Match transform should disregard any persons other than the first name it parses.

With this method you can require either one or both persons to match for the record to match.

Two names; one person

With the *Compare Given_Name1 to Given_Name2* option, you can also compare a record's Given_Name1 data (first name) with the second record's Given_Name2 data (middle name). With this option, the Match transform can correctly identify matching records such as the two partially shown below. Typically, these record pairs represent sons or daughters named for their parents, but known by their middle name.

Record #	First name	Middle name	Last name	Address
170	Leo	Thomas	Smith	225 Pushbutton Dr
198	Tom		Smith	225 Pushbutton Dr

Hyphenated family names

With the *Match on hyphenated family name* option, you can control how matching is performed if a Family_Name (last name) field contains a hyphenated family name (for example, comparing "Smith-Jones" to "Jones"). Choose whether both criteria must have both names to match or just one name that must match for the records to be called a match.

2.16.4.9.3.3.1 Match compound family names

The Approximate Substring Score assists in setting up comparison of compound family names. The Approximate Substring score is assigned to the words that do not match to other words in a compared string.This option loosens some of the requirements of the Substring Adjustment score option in the following ways:

- First words do not have to match exactly.
- The words that do match can use initials and abbreviations adjustments (For example, Rodriguez and RDZ).
- Matching words have to be in the same order, but there can be non-matching words before or after the matching words.
- The Approximate Substring score is assigned the leftover words and spaces in the compared string.

The Approximate Substring option will increase the score for some matches found when using the Substring Matching Score.

Example

When comparing CRUZ RODRIGUEZ and GARCIA CRUZ DE RDZ, the similarity scores are:

- Without setting any adjusments, the Similarity score is 48.
- When you set the Substring adjustment score to 80 and the Abbreviation score to 80, the Similarity score is 66.
- When you set the Approximate substring adjustment score to 80 and the Abbreviation score to 80, the Similarity score is 91.

2.16.4.9.3.4 Numeric data matching

Use the *Numeric words match exactly* option to choose whether data with a mixture of numbers and letters should match exactly. You can also specify *how* this data must match. This option applies most often to address data and custom data, such as a part number.

The numeric matching process is as follows:

1. The string is first broken into words. The word breaking is performed on all punctuation and spacing, and then the words are assigned a numeric attribute. A numeric word is any word that contains at least one number from 0 to 9. For example, 4L is considered a numeric word, whereas FourL is not.
2. Numeric matching is performed according to the option setting that you choose (as described below).

Option values and how they work

Option value	Description
Any_Position	With this value, numeric words must match exactly; however, the position of the word is *not* important. For example: • Street address comparison: "4932 Main St # 101" and "# 101 4932 Main St" are considered a match. • Street address comparison: "4932 Main St # 101" and "# 102 4932 Main St" are *not* considered a match. • Part description: "ACCU 1.4L 29BAR" and "ACCU 29BAR 1.4L" are considered a match.
Same_Position	This value specifies that numeric words must match exactly; however, this option differs from the Any_Position option in that the position of the word *is* important. For example, 608-782-5000 will match 608-782-5000, but it will not match 782-608-5000.
Any_Position_Consider_Punctuation	This value performs word breaking on all punctuation and spaces *except* on the decimal separator (period or comma) so that all decimal numbers are not broken. For example, the string 123.456 is considered a single numeric word as opposed to two numeric words. The position of the numeric word is not important; however, decimal separators do impact the matching process. For example: • Part description: "ACCU 29BAR 1.4L" and "ACCU 1.4L 29BAR" are considered a match. • Part description: "ACCU 1,4L 29BAR" and "ACCU 29BAR 1.4L" are *not* considered a match because there is a decimal indicator between the 1 and the 4 in both cases. • Financial data: "25,435" and "25.435" are not considered a match.
Any_Position_Ignore_Punctuation	This value is similar to the Any_Position_Consider_Punctuation value, except that decimal separators do *not* impact the matching process. For example:

Option value	Description
	- Part description: "ACCU 29BAR 1.4L" and "ACCU 1.4L 29BAR" are considered a match. - Part description: "ACCU 1,4L 29BAR" and "ACCU 29BAR 1.4L" are also considered a match even though there is a decimal indicator between the 1 and the 4. - Part description: "ACCU 29BAR 1.4L" and "ACCU 1.5L 29BAR" are not considered a match.

2.16.4.9.3.5 Blank field matching

In your business rules, you can control how the Match transform treats field comparisons when one or both of the fields compared are blank.

For example, the first name field is blank in the second record shown below. Would you want the Match transform to consider these records matches or no matches? What if the first name field were blank in both records?

Record #1	Record #2
John Doe	_____ Doe
204 Main St	204 Main St
La Crosse WI	La Crosse WI
54601	54601

There are some options in the Match transform that allow you to control the way these are compared. They are:

- Both fields blank operation
- Both fields blank score
- One field blank operation
- One field blank score

Blank field operations

The "operation" options have the following value choices:

Option	Description
Eval	If you choose Eval, the Match transform scores the comparison using the score you enter at the *One field blank score* or *Both fields blank score* option.
Ignore	If you choose Ignore, the score for this field rule does not contribute to the overall weighted score for the record comparison. In other words, the two records shown above could still be considered duplicates, despite the blank field.

Blank field scores

The "Score" options control how the Match transform scores field comparisons when the field is blank in one or both records. You can enter any value from 0 to 100.

To help you decide what score to enter, determine if you want the Match transform to consider a blank field 0 percent similar to a populated field or another blank field, 100 percent similar, or somewhere in between.

Your answer probably depends on what field you're comparing. Giving a blank field a high score might be appropriate if you're matching on a first or middle name or a company name, for example.

Example

Here are some examples that may help you understand how your settings of these blank matching options can affect the overall scoring of records.

One field blank operation for Given_Name1 field set to Ignore

Note that when you set the blank options to Ignore, the Match transform redistributes the contribution allotted for this field to the other criteria and recalculates the contributions for the other fields.

Fields compared	Record A	Record B	% alike	Contribution	Score (per field)
Postcode	54601	54601	100	20 (or 22)	22
Address	100 Water St	100 Water St	100	40 (or 44)	44
Family_Name	Hamilton	Hammilton	94	30 (or 33)	31
Given_Name1	Mary		—	10 (or 0)	—
					Weighted score: **97**

One field blank operation for Given_Name1 field set to Eval; One field blank score set to 0

Fields compared	Record A	Record B	% alike	Contribution	Score (per field)
Postcode	54601	54601	100	20	20
Address	100 Water St	100 Water St	100	40	40
Family_Name	Hamilton	Hammilton	94	30	28
Given_Name1	Mary		0	10	0
					Weighted score: **88**

One field blank operation for Given_Name1 field set to Eval; One field blank score set to 100

Fields compared	Record A	Record B	% alike	Contribution	Score (per field)
Postcode	54601	54601	100	20	20
Address	100 Water St	100 Water St	100	40	40
Family_Name	Hamilton	Hammilton	94	30	28
Given_Name1	Mary		100	10	10
					Weighted score: **98**

2.16.4.9.3.6 Multiple field (cross-field) comparison

In most cases, you use a single field for comparison. For example, Field1 in the first record is compared with Field1 in the second record.

However, there are situations where comparing multiple fields can be useful. For example, suppose you want to match telephone numbers in the Phone field against numbers found in fields used for Fax, Mobile, and Home. Multiple field comparison makes this possible.

When you enable multiple field comparison in the Multiple Field Comparison tab of a match criteria in the Match Editor, you can choose to match selected fields against either all of the selected fields in each record, or against only the same field in each record.

ℹ Note

By default, Match performs multiple field comparison on fields where match standards are used. For example, Person1_Given_Name1 is automatically compared to Person1_Given_Name_Match_Std1-6. Multiple field comparison does not need to be explicitly enabled, and no additional configuration is required to perform multiple field comparison against match standard fields.

2.16.4.9.3.6.1 Comparing selected fields to all selected fields in other records

When you compare each selected field to all selected fields in other records, all fields that are defined in that match criteria are compared against each other.

➜ Remember

“Selected” fields include the criteria field and the other fields you define in the *Additional fields to compare* table.

- If one or more field comparisons meets the settings for Match score, the two rows being compared are considered matches.
- If one or more field comparisons exceeds the No match score, the rule will be considered to pass and any other defined criteria/weighted scoring will be evaluated to determine if the two rows are considered matches.

Example

Example of comparing selected fields to all selected fields in other records

Your input data contains two firm fields.

Row ID	Firm1	Firm2
1	Firstlogic	Postalsoft
2	SAP BusinessObjects	Firstlogic

With the Match score set to 100 and No match score set to 99, these two records are considered matches. Here is a summary of the comparison process and the results.

- First, Row 1 Firm1 (Firstlogic) is compared to Row 2 Firm1 (SAP).
  Normally, the rows would fail this comparison, but with multi-field comparison activated, a No Match decision is not made yet.
- Next, Row 1 Firm2 is compared to Row 2 Firm2 and so on until all other comparisons are made between all fields in all rows. Because Row 1 Firm1 (Firstlogic) and Row 2 Firm2 (Firstlogic) are 100% similar, the two records are considered matches.

2.16.4.9.3.6.2 Comparing selected fields to the same fields in other records

When you compare each selected field to the same field in other records, each field defined in the Multiple Field Comparison tab of a match criteria are compared only to the same field in other records. This sets up, within this criteria, what is essentially an OR condition for passing the criteria. Each field is used to determine a match: If Field_1, Field_2, or Field_3 passes the match criteria, consider the records a match. The No Match score for one field does not automatically fail the criteria when you use multi-field comparison.

➜ Remember

“Selected” fields include the criteria field and the other fields you define in the *Additional fields to compare* table.

Example

Example of comparing selected fields to the same field in other records

Your input data contains a phone, fax, and cell phone field. If any one of these input field's data is the same between thte rows, the records are found to be matches.

Row ID	Phone	Fax	Cell
1	608-555-1234	608-555-0000	608-555-4321
2	608-555-4321	608-555-0000	608-555-1111

With a Match score of 100 and a No match score of 99, the phone and the cell phone number would both fail the match criteria, if defined individually. However, because all three fields are defined in one criteria and the selected records being compared to the same records, the fact that the fax number is 100% similar calls these records a match.

i Note

In the example above, Row 1's cell phone and Row 2's phone would not be considered a match with the selection of the *the same field to other records* option because it only compares within the same field in this case. If this cross-comparison is needed, select the *all selected fields in other records* option instead.

2.16.4.9.3.7 Proximity matching

Proximity matching gives you the ability to match records based on their proximity instead of comparing the string representation of data. You can match on geographic, numeric, and date proximity.

Related Information

Match on Geographic proximity [page 560]
Match on numeric or date proximity [page 561]

2.16.4.9.3.7.1 Match on Geographic proximity

Geographic Proximity finds duplicate records based on geographic proximity, using latitude and longitude information. This is not driving distance, but Geographic distance. This option uses WGS 84 (GPS) coordinates.

The Geographic proximity option can:

- Search on objects within a radial range. This can help a company that wants to send a mailing out to customers within a certain distance from their business location.
- Search on the nearest location. This can help a consumer find a store location closest to their address.

2.16.4.9.3.7.2 Set up Geographic Proximity matching - criteria fields

To select the fields for Geographic Proximity matching, follow these steps:

1. Access the Match Editor, add a new criteria.
2. From Available Criteria, expand *Geographic*.
3. Select *LATITUDE_LONGITUDE*.
   This will make the two criteria fields available for mapping.
4. Map the correct latitude and longitude fields. You must map both fields.

2.16.4.9.3.7.3 Set up Geographic Proximity matching - criteria options

You must have the Latitude and Longitude fields mapped before you can use this option.

To perform geographic proximity matching, follow these steps:

1. From Compare data using, select *Geo Proximity*.

This filters the options under Comparison Rules to show only applicable options.

2. Set the Distance unit option to one of the following:
   - *Miles*
   - *Feet*
   - *Kilometers*
   - *Meters*
3. Enter the *Max Distance* you want to consider for the range.
4. Set the *Max Distance Score*.

> i Note
>
> A distance equal to Max distance will receive a score of Max distance score. Any distance less than the Max distance will receive a proportional score between Max distance score and 100. For example, a proximity of 10 miles will have higher score than a 40 miles.

> i Note
>
> If the data for Max Distance may change from row to row, you should dynamically input the data using the Option_Field_Algorithm_Geo_Proximity_**`<logical_name>`**_Max_Distance field.

Related Information

Reference Guide: Dynamic transform settings [page 1129]
Reference Guide: Match transform input fields [page 1307]

2.16.4.9.3.7.4 Match on numeric or date proximity

The Match Transform's numeric proximity options find duplicates based on numerical closeness of data. You can find duplicates based on numeric values and date values. The following options are available in the Match Criteria Editor Options tab for numeric and date matching:

Numeric difference

Finds duplicates based on the numeric difference for numeric or date values. For example, you can use this option to find duplicates based on date values in a specific range (for example, plus or minus 35 days), regardless of character-based similarity.

Numeric percent difference

Finds duplicates based on the percentage of numeric difference for numeric values. Here are two examples where this might be useful :

- Finance data domain : You can search financial data to find all monthly mortgage payments that are within 5 percent of a given value.
- Product data domain, you can search product data to find all the steel rods that are within 10% tolerance of a given diameter.

2.16.4.10 Post-match processing

2.16.4.10.1 Best record

A key component in most data consolidation efforts is salvaging data from matching records—that is, members of match groups—and posting that data to a best record, or to all matching records.

You can perform these functions by adding a Best Record post-match operation.

Operations happen within match groups

The functions you perform with the Best Record operation involve manipulating or moving data contained in the master records and subordinate records of match groups. Match groups are groups of records that the Match transform has found to be matching, based on the criteria you have created.

A master record is the first record in the Match group. You can control which record this is by using a Group Prioritization operation before the Best Record operation.

Subordinate records are all of the remaining records in a match group.

To help illustrate this use of master and subordinate records, consider the following match group:

Record	Name	Phone	Date	Group rank
#1	John Smith		11 Apr 2001	Master
#2	John Smyth	788-8700	12 Oct 1999	Subordinate
#3	John E. Smith	788-1234	22 Feb 1997	Subordinate
#4	J. Smith	788-3271		Subordinate

Because this is a match group, all of the records are considered matching. As you can see, each record is slightly different. Some records have blank fields, some have a newer date, all have different phone numbers.

A common operation that you can perform in this match group is to move updated data to all of the records in a match group. You can choose to move data to the master record, to all the subordinate members of the match group, or to all members of the match group. The most recent phone number would be a good example here.

Another example might be to salvage useful data from matching records before discarding them. For example, when you run a drivers license file against your house file, you might pick up gender or date-of-birth data to add to your house record.

Post higher priority records first

The operations you set up in the Best Record option group should always start with the highest priority member of the match group (the master) and work their way down to the last subordinate, one at a time. This ensures that data can be salvaged from the higher-priority record to the lower priority record.

So, be sure that your records are prioritized correctly, by adding a Group Prioritization post-match operation before your Best Record operation.

2.16.4.10.1.1 Best record strategies

We provide you with strategies that help you set up some more common best record operation quickly and easily. If none of these strategies fit your needs, you can create a custom best record strategy, using your own Python code.

Best record strategies act as a criteria for taking action on other fields. If the criteria is not met, no action is taken.

Example

In our example of updating a phone field with the most recent data, we can use the Date strategy with the Newest priority to update the master record with the latest phone number in the match group. This latter part (updating the master record with the latest phone number) is the action. You can also update all of the records in the match group (master and all subordinates) or only the subordinates.

⚠ Restriction

The date strategy does not parse the date, because it does not know how the data is formatted. Be sure your data is pre-formatted as YYYYMMDD, so that string comparisons work correctly. You can also do this by setting up a custom strategy, using Python code to parse the date and use a date compare.

2.16.4.10.1.1.1 Custom best record strategies and Python

In the pre-defined strategies for the Best Record strategies, the Match transform auto-generates the Python code that it uses for processing. Included in this code, are variables that are necessary to manage the processing.

Common variables

The common variables you see in the generated Python code are:

Variable	Description
SRC	Signifies the source field.
DST	Signifies the destination field.
RET	Specifies the return value, indicating whether the strategy passed or failed (must be either "T" or "F").

NEWDST and NEWGRP variables

Use the NEWDST and NEWGRP variables to allow the posting of data in your best-record action to be independent of the strategy fields. If you do not include these variables, the strategy field data must also be updated.

Variable	Description
NEWDST	New destination indicator. This string variable will have a value of "T" when the destination record is new or different than the last time the strategy was evaluated and a value of "F" when the destination record has not changed since last time. The NEWDST variable is only useful if you are posting to multiple destinations, such as ALL or SUBS in the *Posting destination* option.
NEWGRP	New group indicator. This string variable will have a value of "T" when the match group is different than the last time the strategy was evaluated and a value of "F" when the match group has not changed since last time.

NEWDST example

The following Python code was generated from a NON_BLANK strategy with options set this way:

Option	Setting
Best record strategy	NON_BLANK
Strategy priority	Priority option not available for the NON_BLANK strategy.
Strategy field	NORTH_AMERICAN_PHONE1_NORTH_AMERICAN_PHONE_STANDARDIZED.
Posting destination	ALL
Post only once per destination	YES

Here is what the Python code looks like.

```
# Setup local temp variable to store updated compare condition
dct = locals()

# Store source and destination values to temporary variables
# Reset the temporary variable when the destination changes
if (dct.has_key('BEST_RECORD_TEMP') and NEWDST.GetBuffer() == u'F'):
  DESTINATION = dct['BEST_RECORD_TEMP']
else:
  DESTINATION =
DST.GetField(u'NORTH_AMERICAN_PHONE1_NORTH_AMERICAN_PHONE_STANDARDIZED')

SOURCE = SRC.GetField(u'NORTH_AMERICAN_PHONE1_NORTH_AMERICAN_PHONE_STANDARDIZED')

if len(SOURCE.strip()) > 0 and len(DESTINATION.strip()) == 0:
  RET.SetBuffer(u'T')
  dct['BEST_RECORD_TEMP'] = SOURCE
else:
  RET.SetBuffer(u'F')
  dct['BEST_RECORD_TEMP'] = DESTINATION

# Delete temporary variables
del SOURCE
del DESTINATION
```

Example

NEWDST and NEWGRP

Suppose you have two match groups, each with three records.

Match group	Records
Match group 1	Record A
	Record B
	Record C
Match group 2	Record D
	Record E
	Record F

Each new destination or match group is flagged with a "T".

NEWGRP	NEWDST	Comparison
(T or F)	(T or F)	
T (New match group)	T (New destination "A")	Record A > Record B
F	F	A > C
F	T (New destination "B")	B > A
F	F	B > C
F	T (New destination "C")	C > A
F	F	C > B
T (New match group)	T (New destination "D")	D > E

NEWGRP (T or F)	NEWDST (T or F)	Comparison
F	F	D > F
F	T (New destination "E")	E > D
F	F	E > F
F	T (New destination "F")	F > D
F	F	F > E

2.16.4.10.1.1.2 To create a pre-defined best record strategy

Be sure to add a Best Record post-match operation to the appropriate match level in the Match Editor. Also, remember to map any pertinent input fields to make them available for this operation.

This procedure allows you to quickly generate the criteria for your best record action. The available strategies reflect common use cases.

1. Enter a name for this Best Record operation.
2. Select a strategy from the *Best record strategy* option.
3. Select a priority from the *Strategy priority* option.
   The selection of values depends on the strategy you chose in the previous step.
4. Select a field from the *Strategy field* drop-down menu.
   The field you select here is the one that acts as a criteria for determining whether a best record action is taken.

Example

The strategy field you choose must contain data that matches the strategy you are creating. For example, if you are using a newest date strategy, be sure that the field you choose contains date data.

2.16.4.10.1.1.3 To create a custom best record strategy

1. Add a best record operation to your Match transform.
2. Enter a name for your best record operation.
3. In the *Best record strategy* option, choose Custom.
4. Choose a field from the *Strategy field* drop-down list.
5. Click the *View/Edit Python* button to create your custom Python code to reflect your custom strategy.
   The Python Editor window appears.

2.16.4.10.1.2 Best record actions

Best record actions are the functions you perform on data if a criteria of a strategy is met.

Example

Suppose you want to update phone numbers of the master record. You would only want to do this if there is a subordinate record in the match group that has a newer date, which signifies a potentially new phone number for that person.

The action you set up would tell the Match transform to update the phone number field in the master record (action) if a newer date in the date field is found (strategy).

2.16.4.10.1.2.1 Sources and destinations

When working with the best record operation, it is important to know the differences between sources and destinations in a best record action.

The source is the field from which you take data and the destination is where you post the data. A source or destination can be either a master or subordinate record in a match group.

Example

In our phone number example, the subordinate record has the newer date, so we take data from the phone field (the source) and post it to the master record (the destination).

2.16.4.10.1.2.2 Posting once or many times per destination

In the Best Record options, you can choose to post to a destination once or many times per action by setting the *Post only once per destination* option.

You may want your best record action to stop after the first time it posts data to the destination record, or you may want it to continue with the other match group records as well. Your choice depends on the nature of the data you're posting and the records you're posting to. The two examples that follow illustrate each case.

If you post only once to each destination record, then once data is posted for a particular record, the Match transform moves on to either perform the next best record action (if more than one is defined) or to the next record.

If you don't limit the action in this way, all actions are performed each time the strategy returns True.

Regardless of this setting, the Match transform always works through the match group members in priority order. When posting to record #1 in the figure below, without limiting the posting to only once, here is what happens:

Match group	Action
Record #1 (master)	
Record #2 (subordinate)	First, the action is attempted using, as a source, that record from among the other match group records that has the highest priority (record #2).
Record #3 (subordinate)	Next, the action is attempted with the next highest priority record (record #3) as the source.
Record #4 (subordinate)	Finally, the action is attempted with the lowest priority record (record #4) as the source.

The results In the case above, record #4 was the last source for the action, and therefore could be a source of data for the output record. However, if you set your best record action to post only once per destination record, here is what happens:

Match group	Action
Record #1 (master)	
Record #2 (subordinate)	First, the action is attempted using, as a source, that record from among the other match group records that has the highest priority (record #2). If this attempt is successful, the Match transform considers this best record action to be complete and moves to the next best record action (if there is one), or to the next output record. If this attempt is not successful, the Match transform moves to the match group member with the next highest priority and attempts the posting operation.
Record #3 (subordinate)	
Record #4 (subordinate)	

In this case, record #2 was the source last used for the best record action, and so is the source of posted data in the output record.

2.16.4.10.1.2.3 To create a best record action

The best record action is the posting of data from a source to a destination record, based on the criteria of your best record strategy.

1. Create a strategy, either pre-defined or custom.
2. Select the record(s) to post to in the *Posting destination* option.
3. Select whether you want to post only once or multiple times to a destination record in the *Post only once per destination* option.
4. In the *Best record action fields table*, choose your source field and destination field.
   When you choose a source field, the *Destination field* column is automatically populated with the same field. You need to change the destination field if this is not the field you want to post your data to.

5. If you want to create a custom best record action, choose Yes in the *Custom* column.
   You can now access the Python editor to create custom Python code for your custom action.

2.16.4.10.1.3 Destination protection

The Best Record and Unique ID operations in the Match transform offer you the power to modify existing records in your data. There may be times when you would like to protect data in particular records or data in records from particular input sources from being overwritten.

The Destination Protection tab in these Match transform operations allow you the ability to protect data from being modified.

2.16.4.10.1.3.1 To protect destination records through fields

1. In the Destination Protection tab, select *Enable destination protection*.
2. Select a value in the *Default destination protection* option drop-down list.

   This value determines whether a destination is protected if the destination protection field does not have a valid value.
3. Select the *Specify destination protection by field* option, and choose a field from the *Destination protection field* drop-down list (or *Unique ID protected field*).

   The field you choose must have a Y or N value to specify the action.

Any record that has a value of Y in the destination protection field will be protected from being modified.

2.16.4.10.1.3.2 To protect destination records based on input source membership

You must add an Input Source operation and define input sources before you can complete this task.

1. In the Destination Protection tab, select *Enable destination protection*.
2. Select a value in the *Default destination protection* option drop-down list.

   This value determines whether a destination (input source) is protected if you do not specifically define the source in the table below.
3. Select the *Specify destination protection by source* option.
4. Select an input source from the first row of the *Source name* column, and then choose a value from the *Destination protected* (or *Unique ID protected*) column.
   Repeat for every input source you want to set protection for. Remember that if you do not specify for every source, the default value will be used.

2.16.4.10.2 Unique ID

A unique ID refers to a field within your data which contains a unique value that is associated with a record or group of records. You could use a unique ID, for example, in your company's internal database that receives updates at some predetermined interval, such as each week, month, or quarter. Unique ID applies to a data record in the same way that a national identification number might apply to a person; for example, a Social Security number (SSN) in the United States, or a National Insurance number (NINO) in the United Kingdom. It creates and tracks data relationships from run to run. With the Unique ID operation, you can set your own starting ID for new key generation, or have it dynamically assigned based on existing data. The Unique ID post-match processing operation also lets you begin where the highest unique ID from the previous run ended.

Unique ID works on match groups

Unique ID doesn't necessarily assign IDs to individual records. It can assign the same ID to every record in a match group (groups of records found to be matches).

If you are assigning IDs directly to a break group, use the *Group number field* option to indicate which records belong together. Additionally, make sure that the records are sorted by group number so that records with the same group number value appear together.

If you are assigning IDs to records that belong to a match group resulting from the matching process, the *Group number field* is not required and should not be used.

> **i Note**
>
> If you are assigning IDs directly to a break group and the *Group number field* is not specified, Match treats the entire data collection as one match group.

2.16.4.10.2.1 Unique ID processing options

The Unique ID post-match processing operation combines the update source information with the master database information to form one source of match group information. The operation can then assign, combine, split, and delete unique IDs as needed. You can accomplish this by using the *Processing operation* option.

Operation	Description
Assign	Assigns a new ID to unique records that don't have an ID or to all members of a group that don't have an ID. In addition, the assign operation copies an existing ID if a member of a match group already has an ID. Each record is assigned a value. ● Records in a match group where one record had an input unique ID will share the value with other records in the match group which had no input value. The first value encountered will be shared. Order affects this; if you have a priority field that can be sequenced

Operation	Description
	using ascending order, place a Prioritization post-match operation prior to the Unique ID operation. • Records in a match group where two or more records had different unique ID input values will each keep their input value. • If all of the records in a match group do not have an input unique ID value, then the next available ID will be assigned to each record in the match group. If the GROUP_NUMBER input field is used, then records with the same group number must appear consecutively in the data collection. Note Use the GROUP_NUMBER input field only when processing a break group that may contain smaller match groups. If the GROUP_NUMBER field is not specified, Unique ID assumes that the entire collection is one group.
AssignCombine	Performs both an Assign and a Combine operation. Each record is assigned a value. • Records that did not have an input unique ID value and are not found to match another record containing an input unique ID value will have the next available ID assigned to it. These are "add" records that could be unique records or could be matches, but not to another record that had previously been assigned a unique ID value. • Records in a match group where one or more records had an input unique ID with the same or different values will share the first value encountered with all other records in the match group. Order affects this; if you have a priority field that can be sequenced using ascending order, place a Prioritization post-match operation prior to the Unique ID operation. If the GROUP_NUMBER input field is used, then records with the same group number must appear consecutively in the data collection. Note Use the GROUP_NUMBER input field only when processing a break group that may contain smaller match groups. If the GROUP_NUMBER field is not specified, Unique ID assumes that the entire collection is one group.
Combine	Ensures that records in the same match group have the same Unique ID. For example, this operation could be used to assign all the members of a household the same unique ID. Specifically, if a household has two members that share a common unique ID, and a third person moves in with a different unique ID, then the Combine operation could be used to assign the same ID to all three members. The first record in a match group that has a unique ID is the record with the highest priority. All other records in the match group are given this record's ID (assuming the record is not protected). The Combine operation does not assign a unique ID to any record that does not al-

Operation	Description
	ready have a unique ID. It only combines the unique ID of records in a match group that already have a unique ID. If the GROUP_NUMBER input field is used, then records with the same group number must appear consecutively in the data collection. Note Use the GROUP_NUMBER input field only when processing a break group that may contain smaller match groups. If the GROUP_NUMBER field is not specified, Unique ID assumes that the entire collection is one group.
Delete	Deletes unique IDs from records that no longer need them, provided that they are not protected from being deleted. If you are using a file and are recycling IDs, this ID is added to the file. When performing a delete, records with the same unique ID should be grouped together. When Match detects that a group of records with the same unique ID is about to be deleted: ● If any of the records are protected, all records in the group are assumed to be protected. ● If recycling is enabled, the unique ID will be recycled only once, even though a group of records had the same ID.
Split	Changes a split group's unique records, so that the records that do not belong to the same match group will have a different ID. The record with the group's highest priority will keep its unique ID. The rest will be assigned new unique IDs. For this operation, you must group your records by unique ID, rather than by match group number. For example: ● Records in a match group where two or more records had different unique ID input values or blank values will each retain their input value, filled or blank depending on the record. ● Records that did not have an input unique ID value and did not match any record with an input unique ID value will have a blank unique ID on output. ● Records that came in with the same input unique ID value that no longer are found as matches have the first record output with the input value. Subsequent records are assigned new unique ID values.

2.16.4.10.2.2 Unique ID protection

The output for the unique ID depends on whether an input field in that record has a value that indicates that the ID is protected.

- If the protected unique ID field is not mapped as an input field, Match assumes that none of the records are protected.
- There are two valid values allowed in this field: Y and N. Any other value is converted to Y.
  A value of N means that the unique ID is not protected and the ID posted on output may be different from the input ID.

a value of Y means that the unique ID is protected and the ID posted on output will be the same as the input ID.

- If the protected unique ID field is mapped as an input field, a value other than N means that the record's input data will be retained in the output unique ID field.

These rules for protected fields apply to all unique ID processing operations.

2.16.4.10.2.3 Unique ID limitations

Because some options in the unique ID operation are based on reading a file or referring to a field value, there may be implications for when you are running a multi-server or real-time server environment and sharing a unique ID file.

- If you are reading from or writing to a file, the unique ID file must be on a shared file system.
- Recycled IDs are used in first-in, first-out order. When Match recycles an ID, it does not check whether the ID is already present in the file. You must ensure that a particular unique ID value is not recycled more than once.

2.16.4.10.2.4 To assign unique IDs using a file

1. In the Unique ID option group, select the *Value from file* option.
2. Set the file name and path in the *File* option.

   This file must be an XML file and must adhere to the following structure:

   ```
   <UniqueIdSession>
       <CurrentUniqueId>477</CurrentUniqueId>
   </UniqueIdSession>
   ```

   > Note
   >
   > The value of 477 is an example of a starting value. However, the value must be 1 or greater.

2.16.4.10.2.5 To assign a unique ID using a constant

Similar to using a file, you can assign a starting unique ID by defining that value.

1. Select the *Constant value* option.
2. Set the *Starting value* option to the desired ID value.

2.16.4.10.2.6 Assign unique IDs using a field

The Field option allows you to send the starting unique ID through a field in your data source or from a User-Defined transform, for example.

The starting unique ID is passed to the Match transform before the first new unique ID is requested. If no unique ID is received, the starting number will default to 1.

> ⚠ Caution
>
> Use caution when using the Field option. The field that you use must contain the unique ID value you want to begin the sequential numbering with. This means that each record you process must contain this field, and each record must have the same value in this field.
>
> For example, suppose the value you use is 100,000. During processing, the first record or match group will have an ID of 100,001. The second record or match group receives an ID of 100,002, and so on.
>
> The value in the first record that makes it to the Match transform contains the value where the incrementing begins.
>
> There is no way to predict which record will make it to the Match transform first (due to sorting, for example); therefore, you cannot be sure which value the incrementing will begin at.

2.16.4.10.2.6.1 To assign unique IDs using a field

1. Select the *Field* option.
2. In the *Starting unique ID field* option, select the field that contains the starting unique ID value.

2.16.4.10.2.7 To assign unique IDs using GUID

You can use Globally Unique Identifiers (GUID) as unique IDs.

Select the *GUID* option.

GUID is also known as the Universal Unique Identifier (UUID). The UUID variation used for unique ID is a time-based 36-character string with the format: `TimeLow-TimeMid-TimeHighAndVersion-ClockSeqAndReservedClockSeqLow-Node`

For more information about UUID, see the Request for Comments (RFC) document.

Related Information

UUID RFC: http://www.ietf.org/rfc/rfc4122.txt

2.16.4.10.2.8 To recycle unique IDs

If unique IDs are dropped during the Delete processing option, you can write those IDs back to a file to be used later.

1. In the Unique ID option group, set the *Processing operation* option to *Delete*.
2. Select the *Value from file* option.
3. Set the file name and path in the *File* option.
4. Set the *Recycle unique IDs* option to *Yes*. This is the same file that you might use for assigning a beginning ID number.

2.16.4.10.2.8.1 Use your own recycled unique IDs

If you have some IDs of your own that you would like to recycle and use in a data flow, you can enter them in the file you want to use for recycling IDs and posting a starting value for your IDs. Enter these IDs in an XML tag of <R></R>. For example:

```
<UniqueIdSession>
    <CurrentUniqueId>477</CurrentUniqueId>
    <R>214</R>
    <R>378</R>

</UniqueIdSession>
```

2.16.4.10.2.9 Destination protection

The Best Record and Unique ID operations in the Match transform offer you the power to modify existing records in your data. There may be times when you would like to protect data in particular records or data in records from particular input sources from being overwritten.

The Destination Protection tab in these Match transform operations allow you the ability to protect data from being modified.

2.16.4.10.2.9.1 To protect destination records through fields

1. In the Destination Protection tab, select *Enable destination protection*.
2. Select a value in the *Default destination protection* option drop-down list.

   This value determines whether a destination is protected if the destination protection field does not have a valid value.
3. Select the *Specify destination protection by field* option, and choose a field from the *Destination protection field* drop-down list (or *Unique ID protected field*).

   The field you choose must have a Y or N value to specify the action.

Any record that has a value of Y in the destination protection field will be protected from being modified.

2.16.4.10.2.9.2 To protect destination records based on input source membership

You must add an Input Source operation and define input sources before you can complete this task.

1. In the Destination Protection tab, select *Enable destination protection*.
2. Select a value in the *Default destination protection* option drop-down list.

   This value determines whether a destination (input source) is protected if you do not specifically define the source in the table below.
3. Select the *Specify destination protection by source* option.
4. Select an input source from the first row of the *Source name* column, and then choose a value from the *Destination protected* (or *Unique ID protected*) column.
   Repeat for every input source you want to set protection for. Remember that if you do not specify for every source, the default value will be used.

2.16.4.10.3 Group statistics

The Group Statistics post-match operation should be added after any match level and any post-match operation for which you need statistics about your match groups or your input sources.

This operation can also counts statistics from logical input sources that you have already identified with values in a field (pre-defined) or from logical sources that you specify in the Input Sources operation.

This operation also allows you to exclude certain logical sources based on your criteria.

> i Note
>
> If you choose to count input source statistics in the Group Statistics operation, Match will also count basic statistics about your match groups.

Group statistics fields

When you include a Group Statistics operation in your Match transform, the following fields are generated by default:

- GROUP_COUNT
- GROUP_ORDER
- GROUP_RANK
- GROUP_TYPE

In addition, if you choose to generate source statistics, the following fields are also generated and available for output:

- SOURCE_COUNT
- SOURCE_ID

- SOURCE_ID_COUNT
- SOURCE_TYPE_ID

Related Information

Reference Guide: Transforms, Match, Output fields [page 1308]

Management Console Guide: Data Quality Reports, Match Source Statistics Summary report [page 2028]

2.16.4.10.3.1 To generate only basic statistics

This task will generate statistics about your match groups, such as how many records in each match group, which records are masters or subordinates, and so on.

1. Add a Group Statistics operation to each match level you want, by selecting *Post Match Processing* in a match level, clicking the *Add* button, and selecting *Group Statistics*.
2. Select *Generate only basic statistics*.
3. Click the *Apply* button to save your changes.

2.16.4.10.3.2 To generate statistics for all input sources

Before you start this task, be sure that you have defined your input sources in the Input Sources operation.

Use this procedure if you are interested in generating statistics for all of your sources in the job.

1. Add a Group Statistics operation to the appropriate match level.
2. Select the *Generate source statistics from input sources* option.

This will generate statistics for all of the input sources you defined in the Input Sources operation.

2.16.4.10.3.3 To count statistics for input sources generated by values in a field

For this task, you do not need to define input sources with the Input Sources operation. You can specify input sources for Match using values in a field.

Using this task, you can generate statistics for all input sources identified through values in a field, or you can generate statistics for a sub-set of input sources.

1. Add a Group Statistics operation to the appropriate match level.
2. Select the *Generate source statistics from source values* option.
3. Select a field from the *Logical source field* drop-down list that contains the values for your logical sources.

4. Enter a value in the *Default logical source value* field.
   This value is used if the logical source field is empty.
5. Select one of the following:

Option	Description
Count all sources	Select to count all sources. If you select this option, you can click the Apply button to save your changes. This task is complete.
Choose sources to count	Select to define a sub-set of input sources to count. If you select this option, you can proceed to step 6 in the task.

6. Choose the appropriate value in the *Default count flag* option.
   Choose Yes to count any source not specified in the *Manually define logical source count flags table*. If you do not specify any sources in the table, you are, in effect, counting all sources.
7. Select *Auto-generate sources* to count sources based on a value in a field specified in the *Predefined count flag field* option.
   If you do not specify any sources in the *Manually define logical source count flags table*, you are telling the Match transform to count all sources based on the (Yes or No) value in this field.
8. In the *Manually define logical source count flags table*, add as many rows as you need to include all of the sources you want to count.

   > Note
   >
   > This is the first thing the Match transform looks at when determining whether to count sources.

9. Add a source value and count flag to each row, to tell the Match transform which sources to count.

> Tip
>
> If you have a lot of sources, but you only want to count two, you could speed up your set up time by setting the *Default count flag* option to No, and setting up the *Manually define logical source count flags table* to count those two sources. Using the same method, you can set up Group Statistics to count everything and not count only a couple of sources.

2.16.4.10.4 Output flag selection

By adding an Output Flag Selection operation to each match level (Post Match Processing) you want, you can flag specific record types for evaluation or routing downstream in your data flow.

Adding this operation generates the Select_Record output field for you to include in your output schema. This output field is populated with a Y or N depending on the type of record you select in the operation.

Your results will appear in the Match Input Source Output Select report. In that report, you can determine which records came from which source or source group and how many of each type of record were output per source or source group.

Record type	Description
Unique	Records that are not members of any match group. No matching records were found. These can be from sources with a normal or special source.

Record type	Description
Single source masters	Highest ranking member of a match group whose members all came from the same source. Can be from normal or special sources.
Single source subordinates	A record that came from a normal or special source and is a subordinate member of a match group.
Multiple source masters	Highest ranking member of a match group whose members came from two or more sources. Can be from normal or special sources.
Multiple source subordinates	A subordinate record of a match group that came from a normal or special source whose members came from two or more sources.
Suppression matches	Subordinate member of a match group that includes a higher-priority record that came from a suppress-type source. Can be from normal or special source.
Suppression uniques	Records that came from a suppress source for which no matching records were found.
Suppression masters	A record that came from a suppress source and is the highest ranking member of a match group.
Suppression subordinates	A record that came from a suppress-type source and is a subordinate member of a match group.

2.16.4.10.4.1 To flag source record types for possible output

1. In the Match editor, for each match level you want, add an Output Flag Select operation.
2. Select the types of records for which you want to populate the Select_Record field with Y.

   The Select_Record output field can then be output from the Match transform for use downstream in the data flow. This is most helpful if you later want to split off suppression matches or suppression masters from your data (by using a Case transform, for example).

2.16.4.11 Association matching

Association matching combines the matching results of two or more match sets (transforms) to find matches that could not be found within a single match set.

You can set up association matching in the Associate transform. This transform acts as another match set in your data flow, from which you can derive statistics.

This match set has two purposes. First, it provides access to any of the generated data from all match levels of all match sets. Second, it provides the overlapped results of multiple criteria, such as name and address, with name and SSN, as a single ID. This is commonly referred to as association matching.

Group numbers

The Associate transform accepts a group number field, generated by the Match transforms, for each match result that will be combined. The transform can then output a new associated group number.

The Associate transform can operate either on all the input records or on one data collection at a time. The latter is needed for real-time support.

> Example
>
> Association example
>
> Say you work at a technical college and you want to send information to all of the students prior to the start of a new school year. You know that many of the students have a temporary local address and a permanent home address.
>
> In this example, you can match on name, address, and postal code in one match set, and match on name and Social Security number (SSN), which is available to the technical college on every student, in another match set.
>
> Then, the Associate transform combines the two match sets to build associated match groups. This lets you identify people who may have multiple addresses, thereby maximizing your one-to-one marketing and mailing efforts.

2.16.4.12 Unicode matching

Unicode matching lets you match Unicode data. You can process any non-Latin1 Unicode data, with special processing for Chinese, Japanese, Korean and Taiwanese (or CJKT) data.

Chinese, Japanese, Korean, and Taiwanese matching

Regardless of the country-specific language, the matching process for CJKT data is the same. For example, the Match transform:

- Considers half-width and full-width characters to be equal.
- Considers native script numerals and Arabic numerals to be equal. It can interpret numbers that are written in native script. This can be controlled with the Convert text to numbers option in the Criteria options group.
- Includes variations for popular, personal, and firm name characters in the referential data.
- Considers firm words, such as Corporation or Limited, to be equal to their variations (Corp. or Ltd.) during the matching comparison process. To find the abbreviations, the transform uses native script variations of the English alphabets during firm name matching.
- Ignores commonly used optional markers for province, city, district, and so on, in address data comparison.
- Intelligently handles variations in a building marker.

Japanese-specific matching capabilities

With Japanese data, the Match transform considers:

- Block data markers, such as chome and banchi, to be equal to those used with hyphenated data.
- Words with or without Okurigana to be equal in address data.
- Variations of *no* marker, *ga* marker, and so on, to be equal.
- Variations of a hyphen or dashed line to be equal.

Unicode match limitations

The Unicode match functionality does not:

- Perform conversions of simplified and traditional Chinese data.
- Match between non-phonetic scripts like kanji, simplified Chinese, and so on.

Route records based on country ID before matching

Before sending Unicode data into the matching process, you must first, as best you can, separate out the data by country to separate match transforms. This can be done by using a Case transform to route country data based on the country ID.

> ➡ Tip
>
> The Match wizard can do this for you when you use the multi-national strategy.

Inter-script matching

Inter-script matching allows you to process data that may contain more than one script by converting the scripts to Latin1. For example one record has Latin1 and other has katakana, or one has Latin and other has Cyrillic. Select *Yes* to enable Inter-script matching. If you prefer to process the data without converting it to Latin1, leave the Inter-script Matching option set *No*. Here are two examples of names matched using inter-script matching:

Name	Can be matched to...
Viktor Ivanov	Виктор Иванов
Takeda Noburu	スッセ フレ

Locale

The Locale option specifies the locale setting for the criteria field. Setting this option is recommended if you plan to use the Text to Numbers feature to specify the locale of the data for locale-specific text-to-number conversion for the purpose of matching. Here a four examples of text-to-number conversion:

Language	Text	Numbers
French	quatre mille cinq cents soixante-sept	4567
German	dreitausendzwei	3002
Italian	cento	100
Spanish	ciento veintisiete	127

For more information on these matching options, see the Match Transform section of the *Reference Guide*

2.16.4.12.1 To set up Unicode matching

1. Use a Case transform to route your data to a Match transform that handles that type of data.
2. Open the *AddressJapan_MatchBatch* Match transform configuration, and save it with a different name.
3. Set the *Match engine* option in the Match transform options to a value that reflects the type of data being processed.
4. Set up your criteria and other desired operations. For more information on Match Criteria options, see the Match Transform section of the *Reference Guide*.

Example

- When possible, use criteria for parsed components for address, firm, and name data, such as Primary_Name or Person1_Family_Name1.
- If you have parsed address, firm, or name data that does not have a corresponding criteria, use the Address_Data1-5, Firm_Data1-3, and Name_Data1-3 criteria.
- For all other data that does not have a corresponding criteria, use the Custom criteria.

2.16.4.13 Phonetic matching

You can use the Double Metaphone or Soundex functions to populate a field and use it for creating break groups or use it as a criteria in matching.

Match criteria

There are instances where using phonetic data can produce more matches when used as a criteria, than if you were to match on other criteria such as name or firm data.

Matching on name field data produces different results than matching on phonetic data. For example:

Name	Comparison score
Smith	72% similar
Smythe	

Name	Phonetic key (primary)	Comparison score
Smith	SMO	100% similar
Smythe	SMO	

Criteria options

If you intend to match on phonetic data, set up the criteria options this way

Option	Value
Compare algorithm	Field
Check for transposed characters	No
Intials adjustment score	0
Substring adjustment score	0
Abbreviation adjustment score	0

Match scores

If you are matching only on the phonetic criteria, set your match score options like this:

Option	Value
Match score	100
No match score	99

If you are matching on multiple criteria, including a phonetic criteria, place the phonetic criteria first in the order of criteria and set your match score options like this:

Option	Value
Match score	101
No match score	99

Blank fields

Remember that when you use break groups, records that have no value are not in the same group as records that have a value (unless you set up matching on blank fields). For example, consider the following two input records:

Mr Johnson	100 Main St	La Crosse	WI	54601
Scott Johnson	100 Main St	La Crosse	WI	54601

After these records are processed by the Data Cleanse transform, the first record will have an empty first name field and, therefore, an empty phonetic field. This means that there cannot be a match, if you are creating break groups. If you are not creating break groups, there cannot be a match if you are not blank matching.

Length of data

The length you assign to a phonetic function output is important. For example:

First name (last name)	Output
S (Johnson)	S
Scott (Johnson)	SKT

Suppose these two records represent the same person. In this example, if you break on more than one character, these records will be in different break groups, and therefore will not be compared.

2.16.4.14 Set up for match reports

We offer many match reports to help you analyze your match results. For more information about these individual reports, see the *Management Console Guide*.

Include Group Statistics in your Match transform

If you are generating the Match Source Statistics Summary report, you must include a Group Statistics operation in your Match and Associate transform(s).

If you want to track your input source statistics, you may want to include an Input Sources operation in the Match transform to define your sources and, in a Group Statistics operation select to generate statistics for your input sources.

> **Note**
>
> You can also generate input source statistics in the Group Statistics operation by defining input sources using field values. You do not necessarily need to include an Input Sources operation in the Match transform.

Turn on report data generation in transforms

In order to generate the data you want to see in match reports other than the Match Source Statistics report, you must set the *Generate report statistics* option to Yes in the Match and Associate transform(s).

By turning on report data generation, you can get information about break groups, which criteria were instrumental in creating a match, and so on.

> ℹ Note
>
> Be aware that turning on the report option can have an impact on your processing performance. It's best to turn off reports after you have thoroughly tested your data flow.

Define names for match sets, levels, and operations

To get the most accurate data in your reports, make sure that you have used unique names in the Match and Associate transforms for your match sets, levels, and each of your pre- and post-match operations, such as Group Prioritization and Group Statistics. This will help you better understand which of these elements is producing the data you are looking at.

Insert appropriate output fields

There are three output fields you may want to create in the Match transform, if you want that data posted in the Match Duplicate Sample report. They are:

- Match_Type
- Group_Number
- Match_Score

2.16.5 Address Cleanse

This section describes how to prepare your data for address cleansing, how to set up address cleansing, and how to understand your output after processing.

Related Information

What is address cleanse? [page 586]
Prepare your input data [page 589]
Determine which transform(s) to use [page 590]
Identify the country of destination [page 593]

Set up the reference files [page 594]
Define the standardization options [page 596]
Supported countries [page 611]

2.16.5.1 How address cleanse works

Address cleanse provides a corrected, complete, and standardized form of your original address data. With the USA Regulatory Address Cleanse transform and for some countries with the Global Address Cleanse transform, address cleanse can also correct or add postal codes. With the DSF2 Walk Sequencer transform, you can add walk sequence information to your data.

What happens during address cleanse?

The USA Regulatory Address Cleanse transform and the Global Address Cleanse transform cleanse your data in the following ways:

- Verify that the locality, region, and postal codes agree with one another. If your data has just a locality and region, the transform usually can add the postal code and vice versa (depending on the country).
- Standardize the way the address line looks. For example, they can add or remove punctuation and abbreviate or spell-out the primary type (depending on what you want).
- Identify undeliverable addresses, such as vacant lots and condemned buildings (USA records only).
- Assign diagnostic codes to indicate why addresses were not assigned or how they were corrected. (These codes are included in the *Reference Guide*).

Reports

The USA Regulatory Address Cleanse transform creates the USPS Form 3553 (required for CASS) and the NCOALink Summary Report. The Global Address Cleanse transform creates reports about your data including the Canadian SERP—Statement of Address Accuracy Report, the Australia Post's AMAS report, and the New Zealand SOA Report.

Related Information

The Address Cleanse transforms [page 587]
Input and output data and field classes [page 588]
Prepare your input data [page 589]
Determine which transform(s) to use [page 590]
Define the standardization options [page 596]
Reference Guide: Supported countries [page 1395]

Reference Guide: Data Quality Appendix, Country ISO codes and assignment engines [page 1395]
Reference Guide: Data Quality Fields, Global Address Cleanse fields [page 1224]
Reference Guide: Data Quality Fields, USA Regulatory Address Cleanse fields [page 1349]

2.16.5.1.1 The transforms

The following table lists the address cleanse transforms and their purpose.

Transform	Description
DSF2 Walk Sequencer	When you perform DSF2 walk sequencing in the software, the software adds delivery sequence information to your data, which you can use with presorting software to qualify for walk-sequence discounts. Remember The software does not place your data in walk sequence order.
Global Address Cleanse and engines	Cleanses your address data from any of the supported countries (not for U.S. certification). You must set up the Global Address Cleanse transform in conjunction with one or more of the Global Address Cleanse engines (Canada, Global Address, or USA). With this transform you can create Canada Post's Software Evaluation and Recognition Program (SERP)—Statement of Address Accuracy Report, Australia Post's Address Matching Processing Summary report (AMAS), and the New Zealand Statement of Accuracy (SOA) report.
USA Regulatory Address Cleanse	Identifies, parses, validates, and corrects USA address data (within the Latin 1 code page) according to the U.S. Coding Accuracy Support System (CASS). Can create the USPS Form 3553 and output many useful codes to your records. You can also run in a non-certification mode as well as produce suggestion lists. Some options include: DPV, DSF2 (augment), eLOT, EWS, GeoCensus, LACSLink, NCOALink, RDI, SuiteLink, suggestion lists (not for certification), and Z4Change.
Global Suggestion Lists	Offers suggestions for possible address matches for your USA, Canada, and global address data. This transform is usually used for real time processing and does not standardize addresses. Use a Country ID transform before this transform in the data flow. Also, if you want to standardize your address data, use the Global Address Cleanse transform after the Global Suggestion Lists transform in the data flow.
Country ID	Identifies the country of destination for the record and outputs an ISO code. Use this transform before the Global Suggestion Lists transform in your data flow. (It is not necessary to place the Country ID transform before the Global Address Cleanse or the USA Regulatory Address Cleanse transforms.)

2.16.5.1.2 Input and output data and field classes

Input data

The address cleanse transforms accept discrete, multiline, and hybrid address line formats.

Output data

There are two ways that you can set the software to handle output data. Most use a combination of both.

Concept	Description
Multiline	The first method is useful when you want to keep output address data in the same arrangement of fields as were input. The software applies intelligent abbreviation, when necessary, to keep the data within the same field lengths. Data is capitalized and standardized according to the way you set the standardization style options.
Discrete	The second method is useful when you want the output addresses broken down into smaller elements than you input. Also, you can retrieve additional fields created by the software, such as the error/status code. The style of some components is controlled by the standardization style options; most are not. The software does not apply any intelligent abbreviation to make components fit your output fields.

When you set up the USA Regulatory Address Cleanse transform or the Global Address Cleanse transform, you can include output fields that contain specific information:

Generated Field Address Class	Generated Field Class
Delivery	Parsed: Contains the parsed input with some standardization applied. The fields subjected to standardization are locality, region, and postcode. Best: Contains the parsed data when the address is unassigned or the corrected data for an assigned address. Corrected: Contains the assigned data after directory lookups and will be blank if the address is not assigned.
Dual	Parsed, Best, and Corrected: Contain the DUAL address details that were available on input.
Official	Parsed: Contains the parsed input with some standardization applied. Best: Contains the information from directories defined by the Postal Service when an address is assigned. Contains the parsed input when an address is unassigned.

Generated Field Address Class	Generated Field Class
	Corrected: Contains the information from directories defined by the Postal Service when an address is assigned and will be blank if the address is not assigned.

2.16.5.2 Prepare your input data

Before you start address cleansing, you must decide which kind of address line format you will input. Both the USA Regulatory Address Cleanse transform and the Global Address Cleanse transform accept input data in the same way.

⚠ Caution

The USA Regulatory Address Cleanse Transform does not accept Unicode data. If an input record has characters outside the Latin1 code page (character value is greater than 255), the USA Regulatory Address Cleanse transform will not process that data. Instead, the input record is sent to the corresponding standardized output field without any processing. No other output fields (component, for example) will be populated for that record. If your Unicode database has valid U.S. addresses from the Latin1 character set, the USA Regulatory Address Cleanse transform processes as usual.

Accepted address line formats

The following tables list the address line formats: multiline, hybrid, and discrete.

ℹ Note

For all multiline and hybrid formats listed, you are not required to use all the multiline fields for a selected format (for example Multiline1-12). However, you must start with Multiline1 and proceed consecutively. You cannot skip numbers, for example, from Multiline1 to Multiline3.

Multiline and multiline hybrid formats				
Example 1	**Example 2**	**Example 3**	**Example 4**	**Example 5**
Multiline1	Multiline1	Multiline1	Multiline1	Multiline1
Multiline2	Multiline2	Multiline2	Multiline2	Multiline2
Multiline3	Multiline3	Multiline3	Multiline3	Multiline3
Multiline4	Multiline4	Multiline4	Multiline4	Multiline4
Multiline5	Multiline5	Locality3	Multiline5	Multiline5
Multiline6	Multiline6	Locality2	Locality2	Multiline6
Multiline7	Multiline7	Locality1	Locality1	Locality1

Multiline and multiline hybrid formats				
Example 1	**Example 2**	**Example 3**	**Example 4**	**Example 5**
Multiline8	Lastline	Region1	Region1	Region1
Country (Optional)	Country (Optional)	Postcode (Global) or Postcode1 (USA Reg.)	Postcode (Global) or Postcode1 (USA Reg.)	Postcode (Global) or Postcode1 (USA Reg.)
		Country (Optional)	Country (Optional)	Country (Optional)

Discrete line formats			
Example 1	**Example 2**	**Example 3**	**Example 4**
Address_Line	Address_Line	Address_Line	Address_Line
Lastline	Locality3 (Global)	Locality2	Locality1
Country (Optional)	Locality2	Locality1	Region1
	Locality1	Region1	Postcode (Global) or Postcode1 (USA Reg.)
	Region1	Postcode (Global) or Postcode1 (USA Reg.)	Country (Optional)
	Postcode (Global) or Postcode1 (USA Reg.)	Country (Optional)	
	Country (Optional)		

2.16.5.3 Determine which transform(s) to use

You can choose from a variety of address cleanse transforms based on what you want to do with your data. There are transforms for cleansing global and/or U.S. address data, cleansing based on USPS regulations, using business rules to cleanse data and cleansing global address data transactionally.

Related Information

Cleanse global address data [page 591]
Cleanse U.S. data only [page 591]
Cleanse USA data and global data [page 592]
Cleanse address data using multiple business rules [page 592]
Cleanse your address data transactionally [page 593]

2.16.5.3.1 Cleanse global address data

To cleanse your address data for any of the software-supported countries (including Canada for SERP, Software Evaluation and Recognition Program, certification and Australia for AMAS, Address Matching Approval System, certification), use the Global Address Cleanse transform in your project with one or more of the following engines:

- Canada
- Global Address
- USA

> ➜ Tip
>
> Cleansing U.S. data with the USA Regulatory Address Cleanse transform is usually faster than with the Global Address Cleanse transform and USA engine. This scenario is usually true even if you end up needing both transforms.

You can also use the Global Address Cleanse transform with the Canada, USA, Global Address engines in a real time data flow to create suggestion lists for those countries.

Start with a sample transform configuration

The software includes a variety of Global Address Cleanse sample transform configurations (which include at least one engine) that you can copy to use for a project.

Related Information

Supported countries [page 611]
Cleanse U.S. data and global data [page 592]
Reference Guide: Transforms, Transform configurations [page 1123]

2.16.5.3.2 Cleanse U.S. data only

To cleanse U.S. address data, use the USA Regulatory Address Cleanse transform for the best results. With this transform, and with DPV, LACSLink, and SuiteLink enabled, you can produce a CASS-certified mailing and produce a USPS Form 3553. If you do not intend to process CASS-certified lists, you should still use the USA Regulatory Address Cleanse transform for processing your U.S. data. Using the USA Regulatory Address Cleanse transform on U.S. data is more efficient than using the Global Address Cleanse transform.

With the USA Regulatory Address Cleanse transform you can add additional information to your data such as DSF2, EWS, eLOT, NCOALink, and RDI. And you can process records one at a time by using suggestion lists.

Start with a sample transform configuration

The software includes a variety of USA Regulatory Address Cleanse sample transform configurations that can help you set up your projects.

Related Information

Reference Guide: Transforms, Data Quality transforms, Transform configurations [page 1123]
Introduction to suggestion lists [page 676]

2.16.5.3.3 Cleanse U.S. data and global data

What should you do when you have U.S. addresses that need to be certified and also addresses from other countries in your database? In this situation, you should use both the Global Address Cleanse transform and the USA Regulatory Address Cleanse transform in your data flow.

> ➡ Tip
>
> Even if you are not processing U.S. data for USPS certification, you may find that cleansing U.S. data with the USA Regulatory Address Cleanse transform is faster than with the Global Address Cleanse transform and USA engine.

2.16.5.3.4 Cleanse address data using multiple business rules

When you have two addresses intended for different purposes (for example, a billing address and a shipping address), you should use two of the same address cleanse transforms in a data flow.

One or two engines?

When you use two Global Address Cleanse transforms for data from the same country, they can share an engine. You do not need to have two engines of the same kind. If you use one engine or two, it does not affect the overall processing time of the data flow.

In this situation, however, you may need to use two separate engines (even if the data is from the same country). Depending on your business rules, you may have to define the settings in the engine differently for a billing address or for a shipping address. For example, in the Standardization Options group, the Output Country Language option can convert the data used in each record to the official country language or it can preserve the language used in each record. For example, you may want to convert the data for the shipping address but preserve the data for the billing address.

2.16.5.3.5 Cleanse your address data transactionally

The Global Suggestion Lists transform, best used in transactional projects, is a way to complete and populate addresses with minimal data, or it can offer suggestions for possible matches. For example, the Marshall Islands and the Federated States of Micronesia were recently removed from the USA Address directory. Therefore, if you previously used the USA engine, you'll now have to use the Global Address engine. The Global Suggestion Lists transform can help identify that these countries are no longer in the USA Address directory.

This easy address-entry system is ideal in call center environments or any transactional environment where cleansing is necessary at the point of entry. It's also a beneficial research tool when you need to manage bad addresses from a previous batch process.

Place the Global Suggestion Lists transform after the Country ID transform and before a Global Address Cleanse transform that uses a Global Address, Canada, and/or USA engine.

Integrating functionality

Global Suggestion Lists functionality is designed to be integrated into your own custom applications via the Web Service. If you are a programmer looking for details about how to integrate this functionality, see "Integrate Global Suggestion Lists" in the *Integrator Guide*.

Start with a sample transform configuration

Data Quality includes a Global Suggestion Lists sample transform that can help you when setting up a project.

Related Information

Introduction to suggestion lists [page 676]

2.16.5.4 Identify the country of destination

The Global Address Cleanse transform includes Country ID processing. Therefore, you do not need to place a Country ID transform before the Global Address Cleanse transform in your data flow.

In the Country ID Options option group of the Global Address Cleanse transform, you can define the country of destination or define whether you want to run Country ID processing.

Constant country

If all of your data is from one country, such as Australia, you do not need to run Country ID processing or input a discrete country field. You can tell the Global Address Cleanse transform the country and it will assume all records are from this country (which may save processing time).

Assign default

You'll want to run Country ID processing if you are using two or more of the engines and your input addresses contain country data (such as the two-character ISO code or a country name), or if you are using only one engine and your input source contains many addresses that cannot be processed by that engine. Addresses that cannot be processed are not sent to the engine. The transform will use the country you specify in this option group as a default.

Related Information

To set a constant country [page 612]
Set a default country [page 611]

2.16.5.5 Set up the reference files

The USA Regulatory Address Cleanse transform and the Global Address Cleanse transform and engines rely on directories (reference files) in order to cleanse your data.

Directories

To correct addresses and assign codes, the address cleanse transforms rely on databases called postal directories. The process is similar to the way that you use the telephone directory. A telephone directory is a large table in which you look up something you know (a person's name) and read off something you don't know (the phone number).

In the process of looking up the name in the phone book, you may discover that the name is spelled a little differently from the way you thought. Similarly, the address cleanse transform looks up street and city names in the postal directories, and it corrects misspelled street and city names and other errors.

Sometimes it doesn't work out. We've all had the experience of looking up someone and being unable to find their listing. Maybe you find several people with a similar name, but you don't have enough information to tell which listing was the person you wanted to contact. This type of problem can prevent the address cleanse transforms from fully correcting and assigning an address.

Besides the basic address directories, there are many specialized directories that the USA Regulatory Address Cleanse transform uses:

- DPV®
- DSF2®
- Early Warning System (EWS)
- eLOT®
- GeoCensus
- LACSLink®
- NCOALink®
- RDI™
- SuiteLink™
- Z4Change

These features help extend address cleansing beyond the basic parsing and standardizing.

Define directory file locations

You must tell the transform or engine where your directory (reference) files are located in the Reference Files option group. Your system administrator should have already installed these files to the appropriate locations based on your company's needs.

⚠ Caution

Incompatible or out-of-date directories can render the software unusable. The system administrator must install weekly, monthly or bimonthly directory updates for the USA Regulatory Address Cleanse Transform; monthly directory updates for the Australia and Canada engines; and quarterly directory updates for the Global Address engine to ensure that they are compatible with the current software.

Substitution files

If you start with a sample transform, the Reference Files options are filled in with a substitution variable (such as $$RefFilesAddressCleanse) by default. These substitution variables point to the reference data folder of the software directory by default.

You can change that location by editing the substitution file associated with the data flow. This change is made for every data flow that uses that substitution file.

Related Information

DPV [page 618]

DSF2 [page 640]

DSF2 walk sequencing [page 645]

Early Warning System (EWS) [page 667]

eLOT [page 666]

GeoCensus [page 671]

LACSLink [page 629]

NCOALink (USA Regulatory Address Cleanse) [page 650]

RDI (USA Regulatory Address Cleanse) [page 668]

SuiteLink (USA Regulatory Address Cleanse) [page 637]

Z4Change [page 675]

2.16.5.5.1 View directory expiration dates in the trace log

You can view directory expiration information for a current job in the trace log. To include directory expiration information in the trace log, perform the following steps.

1. Right click on the applicable job icon in Designer and select *Execute*.
2. In the *Execution Properties* window, open the *Execution Options* tab (it should already be open by default).
3. Select *Print all trace messages*.

Related Information

Using Logs [page 426]

2.16.5.6 Define the standardization options

Standardization changes the way the data is presented after an assignment has been made. The type of change depends on the options that you define in the transform. These options include casing, punctuation, sequence, abbreviations, and much more. It helps ensure the integrity of your databases, makes mail more deliverable, and gives your communications with customers a more professional appearance.

For example, the following address was standardized for capitalization, punctuation, and postal phrase (route to RR).

Input	Output

Multiline1 = route 1 box 44a
Multiline2 = stodard wisc
	Address_Line = RR 1 BOX 44A
	Locality1 = STODDARD
	Region1 = WI
	Postcode1 = 54658

Global Address Cleanse transform

In the Global Address Cleanse transform, you set the standardization options in the Standardization Options option group.

You can standardize addresses for all countries and/or for individual countries (depending on your data). For example, you can have one set of French standardization options that standardize addresses within France only, and another set of Global standardization options that standardize all other addresses.

USA Regulatory Address Cleanse transform

If you use the USA Regulatory Address Cleanse transform, you set the standardization options on the *Options* tab in the Standardization Options section.

Related Information

Reference Guide: Transforms, Global Address Cleanse transform options (Standardization options) [page 1199]
Reference Guide: Transforms, USA Regulatory Address Cleanse (Standardization options) [page 1336]

2.16.5.7 Process Japanese addresses

The Global Address Cleanse transform's Global Address engine parses Japanese addresses. The primary purpose of this transform and engine is to parse and normalize Japanese addresses for data matching and cleansing applications.

> i Note
>
> The Japan engine only supports kanji and katakana data. The engine does not support Latin data.

A significant portion of the address parsing capability relies on the Japanese address database. The software has data from the Ministry of Public Management, Home Affairs, Posts and Telecommunications (MPT) and additional data sources. The enhanced address database consists of a regularly updated government database that includes regional postal codes mapped to localities.

Related Information

Standard Japanese address format [page 598]
Special Japanese address formats [page 602]
Sample Japanese address [page 604]

2.16.5.7.1 Standard Japanese address format

A typical Japanese address includes the following components.

Address component	Japanese	English	Output field(s)
Postal code	〒 654-0153	654-0153	Postcode_Full
Prefecture	兵庫県	Hyogo-ken	Region1_Full
City	神戸市	Kobe-shi	Locality1_Full
Ward	須磨区	Suma-ku	Locality2_Full
District	南落合	Minami Ochiai	Locality3_Full
Block number	1 丁目	1 chome	Primary_Name_Full1
Sub-block number	25 番地	25 banchi	Primary_Name_Full2
House number	2 号	2 go	Primary_Number_Full

An address may also include building name, floor number, and room number.

Postal code

Japanese postal codes are in the *nnn-nnnn* format. The first three digits represent the area. The last four digits represent a location in the area. The possible locations are district, sub-district, block, sub-block, building, floor, and company. Postal codes must be written with Arabic numbers. The post office symbol 〒 is optional.

Before 1998, the postal code consisted of 3 or 5 digits. Some older databases may still reflect the old system.

Prefecture

Prefectures are regions. Japan has forty-seven prefectures. You may omit the prefecture for some well known cities.

City

Japanese city names have the suffix 市 (-shi). In some parts of the Tokyo and Osaka regions, people omit the city name. In some island villages, they use the island name with a suffix 島 (-shima) in place of the city name. In some rural areas, they use the county name with suffix 郡 (-gun) in place of the city name.

Ward

A city is divided into wards. The ward name has the suffix 区(-ku). The ward component is omitted for small cities, island villages, and rural areas that don't have wards.

District

A ward is divided into districts. When there is no ward, the small city, island village, or rural area is divided into districts. The district name may have the suffix 町 (-cho/-machi), but it is sometimes omitted. 町 has two possible pronunciations, but only one is correct for a particular district.

In very small villages, people use the village name with suffix 村 (-mura) in place of the district.

When a village or district is on an island with the same name, the island name is often omitted.

Sub-district

Primarily in rural areas, a district may be divided into sub-districts, marked by the prefix 字 (aza-). A sub-district may be further divided into sub-districts that are marked by the prefix 小字 (koaza-), meaning small aza. koaza may be abbreviated to aza. A sub-district may also be marked by the prefix 大字 (oaza-), which means large aza. Oaza may also be abbreviated to aza.

Here are the possible combinations:

- oaza
- aza
- oaza and aza
- aza and koaza
- oaza and koaza

> Note
>
> The characters 大字(oaza-), 字(aza-), and 小字 (koaza-) are frequently omitted.

Sub-district parcel

A sub-district aza may be divided into numbered sub-district parcels, which are marked by the suffix 部 (-bu), meaning *piece*. The character 部 is frequently omitted.

Parcels can be numbered in several ways:

- Arabic numbers (1, 2, 3, 4, and so on)
  石川県七尾市松百町８部３番地１号
- Katakana letters in iroha order (イ, ロ, ハ, ニ, and so on)

- 石川県小松市里川町ナ部23番地
- Kanji numbers, which is very rare (甲, 乙, 丙, 丁, and so on)
  愛媛県北条市上難波甲部311番地

Sub-division

A rural district or sub-district (oaza/aza/koaza) is sometimes divided into sub-divisions, marked by the suffix 地割 (-chiwari) which means division of land. The optional prefix is 第 (dai-)

The following address examples show sub-divisions:

岩手県久慈市旭町10地割1番地

岩手県久慈市旭町第10地割1番地

Block number

A district is divided into blocks. The block number includes the suffix 丁目 (-chome). Districts usually have between 1 and 5 blocks, but they can have more. The block number may be written with a Kanji number. Japanese addresses do not include a street name.

東京都渋谷区道玄坂２丁目２５番地１２号

東京都渋谷区道玄坂二丁目２５番地１２号

Sub-block number

A block is divided into sub-blocks. The sub-block name includes the suffix 番地 (-banchi), which means numbered land. The suffix 番地 (-banchi) may be abbreviated to just 番 (-ban).

House number

Each house has a unique house number. The house number includes the suffix 号 (-go), which means number.

Block, sub-block, and house number variations

Block, sub-block, and house number data may vary.

Dashes

The suffix markers 丁目(chome), 番地 (banchi), and 号(go) may be replaced with dashes.

東京都文京区湯島 2 丁目 18 番地 12 号

東京都文京区湯島 2-18-12

Sometimes block, sub-block, and house number are combined or omitted.

東京都文京区湯島 2 丁目 18 番 12 号

東京都文京区湯島 2 丁目 18 番地 12

東京都文京区湯島 2 丁目 18-12

No block number

Sometimes the block number is omitted. For example, this ward of Tokyo has numbered districts, and no block numbers are included. 二番町 means district number 2.

東京都 千代田区 二番町 9 番地 6 号

Building names

Names of apartments or buildings are often included after the house number. When a building name includes the name of the district, the district name is often omitted. When a building is well known, the block, sub-block, and house number are often omitted. When a building name is long, it may be abbreviated or written using its acronym with English letters.

The following are the common suffixes:

Suffix	Romanized	Translation
ビルディング	birudingu	building
ビルヂング	birudingu	building
ビル	biru	building
センター	senta-	center
プラザ	puraza	plaza
パーク	pa-ku	park
タワー	tawa-	tower
会館	kaikan	hall

Suffix	Romanized	Translation
棟	tou	building (unit)
庁舎	chousha	government office building
マンション	manshon	condominium
団地	danchi	apartment complex
アパート	apa-to	apartment
荘	sou	villa
住宅	juutaku	housing
社宅	shataku	company housing
官舎	kansha	official residence

Building numbers

Room numbers, apartment numbers, and so on, follow the building name. Building numbers may include the suffix 号室 (-goshitsu). Floor numbers above ground level may include the suffix 階 (-kai) or the letter F. Floor numbers below ground level may include the suffix 地下<n>階 (chika <n> kai) or the letters B<n>F (where <n> represents the floor number). An apartment complex may include multiple buildings called Building A, Building B, and so on, marked by the suffix 棟 (-tou).

The following address examples include building numbers.

- Third floor above ground
  東京都千代田区二番町９番地６号　バウエプタ３Ｆ
- Second floor below ground
  東京都渋谷区道玄坂 2-25-12 シティバンク地下２階
- Building A Room 301
  兵庫県神戸市須磨区南落合 1-25-10　須磨パークヒルズ　A 棟 301 号室
- Building A Room 301
  兵庫県神戸市須磨区南落合 1-25-10　須磨パークヒルズ　A-301

2.16.5.7.2 Special Japanese address formats

Hokkaido regional format

The Hokkaido region has two special address formats:

- super-block

- numbered sub-districts

Super-block

A special super-block format exists only in the Hokkaido prefecture. A super-block, marked by the suffix 条 (-joh), is one level larger than the block. The super-block number or the block number may contain a directional 北 (north), 南 (south), 東 (east), or 西 (west). The following address example shows a super-block 4 Joh.

北海道札幌市西区二十四軒４条４丁目１３番地７号

Numbered sub-districts

Another Hokkaido regional format is numbered sub-district. A sub-district name may be marked with the suffix 線 (-sen) meaning number instead of the suffix 字 (-aza). When a sub-district has a 線 suffix, the block may have the suffix 号 (-go), and the house number has no suffix.

The following is an address that contains first the sub-district 4 sen and then a numbered block 5 go.

北海道旭川市西神楽４線５号３番地１１

Accepted spelling

Names of cities, districts and so on can have multiple accepted spellings because there are multiple accepted ways to write certain sounds in Japanese.

Accepted numbering

When the block, sub-block, house number or district contains a number, the number may be written in Arabic or Kanji. For example, 二番町 means district number 2, and in the following example it is for Niban-cho.

東京都千代田区二番町九番地六号

P.O. Box addresses

P.O. Box addresses contain the postal code, Locality1, prefecture, the name of the post office, the box marker, and the box number.

Note

The Global Address Cleanse transform recognizes P.O. box addresses that are located in the Large Organization Postal Code (LOPC) database only.

The address may be in one of the following formats:

- Prefecture, Locality1, post office name, box marker (私書箱), and P.O. box number.
- Postal code, prefecture, Locality1, post office name, box marker (私書箱), and P.O. box number.

The following address example shows a P.O. Box address:

The Osaka Post Office Box marker #1

大阪府大阪市大阪支店私書箱1号

Large Organization Postal Code (LOPC) format

The Postal Service may assign a unique postal code to a large organization, such as the customer service department of a major corporation. An organization may have up to two unique postal codes depending on the volume of mail it receives. The address may be in one of the following formats:

- Address, company name
- Postal code, address, company name

The following is an example of an address in a LOPC address format.

100-8798 東京都千代田区霞が関1丁目3-2 日本郵政 株式会社

2.16.5.7.3 Sample Japanese address

This address has been processed by the Global Address Cleanse transform and the Global Address engine.

Input
0018521 北海道札幌市北区北十条西1丁目12番地3号創生ビル1階101号室札幌私書箱センター

Address-line fields	
Primary_Name1	1
Primary_Type1	丁目
Primary_Name2	12
Primary_Type2	番地
Primary_Number	3
Primary_Number_Description	号
Building_Name1	創生ビル
Floor_Number	1

Address-line fields	
Floor_Description	階
Unit_Number	101
Unit_Description	号室
Primary_Address	1丁目12番地3号
Secondary_Address	創生ビル1階101号室
Primary_Secondary_Address	1丁目12番地3号 創生ビル1階101号室

Last line fields	
Country	日本
ISO_Country_Code_3Digit	392
ISO_Country_Code_2Char	JP
Postcode1	001
Postcode2	8521
Postcode_Full	001-8521
Region1	北海
Region1_Description	道
Locality1_Name	札幌
Locality1_Description	市
Locality2_Name	北
Locality2_Description	区
Locality3_Name	北十条西
Lastline	001-8521 北海道 札幌市 北区 北十条西

Firm	
Firm	札幌私書箱センター

Non-parsed fields	
Status_Code	S0000

Non-parsed fields	
Assignment_Type	F
Address_Type	S

2.16.5.8 Process Chinese addresses

The Global Address Cleanse transform's Global Address engine parses Chinese addresses. The primary purpose of this transform and engine is to parse and normalize addresses for data matching and cleansing applications.

2.16.5.8.1 Chinese address format

Chinese addresses are written starting with the postal code, followed by the largest administrative region (for example, province), and continue to the smallest unit (for example, room number and mail receiver). When people send mail between different prefectures, they often include the largest administrative region in the address. The addresses contain detailed information about where the mail will be delivered. Buildings along the street are numbered sequentially, sometimes with odd numbers on one side and even numbers on the other side. In some instances both odd and even numbers are on the same side of the street.

Postal Code

In China, the postal code is a six-digit number to identify the target deliver point of the address, and often has the prefix **邮编**

Country

"**中华人民共和国**" (People's Republic of China) is the full name of China. "**中国**" (PRC) is often used as an abbreviation of the country name. For mail delivered within China, the domestic addresses often omit the country name of the target address.

Province

In China, "provinces" are similar to a "state" in the United States. China has 34 province-level divisions, including:

- *Provinces*(**省** shěng)
- *Autonomous regions*(**自治区** zìzhìqū)

- *Municipalities*(直辖市 zhíxiáshì)
- *Special administrative regions*(特别行政区 tèbié xíngzhèngqū)

Prefecture

Prefecture-level divisions are the second level of the administrative structure, including:

- *Prefectures* (地区 dìqū)
- *Autonomous prefectures* (自治州 zìzhìzhōu)
- *Prefecture-level cities* (地级市 dìjíshì)
- *Leagues* (盟 méng)

County

The county is the sub-division of Prefecture, including:

- *Counties* (县 xiàn)
- *Autonomous counties* (自治县 zìzhìxiàn)
- *County-level cities*(县级市 xiànjíshì)
- *Districts* (市辖区 shìxiáqū)
- *Banners* (旗 qí)
- *Autonomous banners* (自治旗 zìzhìqí)
- *Forestry areas* (林区 línqū)
- *Special districts* (特区 tèqū)

Township

Township level division includes:

- *Townships* (乡 xiāng)
- *Ethnic townships* (民族乡 mínzúxiāng)
- *Towns* (镇 zhèn)
- *Subdistricts* (街道办事处 jiēdàobànshìchù)
- *District public offices* (区公所 qūgōngsuǒ)
- *Sumu*(苏木 sūmù)
- *Ethnic sumu* (民族苏木 mínzúsūmù)

Village

Village includes:

- *Neighborhood committees*(社区居民委员会 jūmínwěiyuánhùi)
- *Neighborhoods or communities* (社区)
- *Village committees*(村民委员会 cūnmínwěiyuánhùi) or *Village groups* (村民小组 cūnmínxiǎozǔ)
- *Administrative villages*(行政村 xíngzhèngcūn)

Street information

Specifies the delivery point where the mail receiver can be found within it. In China, the street information often has the form of street (road) name -> House number. For example, 上海市浦东新区晨晖路 1001 号

- *Street name*: The street name is usually followed by one of these suffixes 路, 大道, 街, 大街, and so on.
- *House number*:
  The house number is followed by the suffix 号, the house number is a unique number within the street/road.

Residential community

In China, the residential community might be used for mail delivery. Especially for some famous residential communities in major cities, the street name and house number might be omitted. The residential community does not have a naming standard and it is not strictly required to be followed by a typical marker. However, it is often followed by the typical suffixes, such as 新村, 小区, and so on (for example, 新村, 小区).

Building name

Building is often followed by the building marker, such as 大厦, 大楼 and so on, though it is not strictly required (for example, 中华大厦). Building name in the residential communities is often represented by a number with a suffix of 号，幢 and so on (for example: 上海市浦东新区晨晖路 100 弄 10 号 101 室).

Common metro address

This address includes the District name, which is common for metropolitan areas in major cities.

Address component	Chinese	English	Output field
Postcode	510030	510030	Postcode_Full
Country	中国	China	Country
Province	广东省	Guangdong Province	Region1_Full
City name	广州市	Guangzhou City	Locality1_Full

Address component	Chinese	English	Output field
District name	越秀区	Yuexiu District	Locality2_Full
Street name	西湖路	Xihu Road	Primary_Name_Full1
House number	99 号	No. 99	Primary_Number_Full

Rural address

This address includes the Village name, which is common for rural addresses.

Address component	Chinese	English	Output field
Postcode	5111316	5111316	Postcode_Full
Country	中国	China	Country
Province	广东省	Guangdong Province	Region1_Full
City name	广州市	Guangzhou City	Locality1_Full
County-level City name	增城市	Zengcheng City	Locality2_Full
Town name	荔城镇	Licheng Town	Locality3_Full
Village name	联益村	Lianyi Village	Locality4_Full
Street name	光大路	Guangda Road	Primary_Name_Full1
House number	99 号	No. 99	Primary_Number_Full

2.16.5.8.2 Sample Chinese address

This address has been processed by the Global Address Cleanse transform and the Global Address engine.

Input
510830 广东省广州市花都区赤坭镇广源路 1 号星辰大厦 8 层 809 室

Address-Line fields	
Primary_Name1	广源
Primary_Type1	路
Primary_Number	1

Address-Line fields	
Primary_Number_Description	号
Building_Name1	星辰大厦
Floor_Number	8
Floor_Description	层
Unit_Number	809
Unit_Description	室
Primary_Address	广源路 1 号
Secondary_Address	星辰大厦 8 层 809 室
Primary_Secondary_Address	广源路 1 号星辰大厦 8 层 809 室

Lastline fields	
Country	中国
Postcode_Full	510168
Region1	广东
Region1_Description	省
Locality1_Name	广州
Locality1_Description	市
Locality2_Name	花都
Locality2_Description	区
Locality3_Name	赤坭
Locality3_Description	镇
Lastline	510830 广东省广州市花都区赤坭镇

Non-parsed fields	
Status_Code	S0000
Assignment_Type	S
Address_Type	S

2.16.5.9 Supported countries (Global Address Cleanse)

There are several countries supported by the Global Address Cleanse transform. The level of correction varies by country and by the engine that you use. Complete coverage of all addresses in a country is not guaranteed.

For the Global Address engine, country support depends on which sets of postal directories you have purchased.

For Japan, the assignment level is based on data provided by the Ministry of Public Management Home Affairs, Posts and Telecommunications (MPT).

During Country ID processing, the transform can identify many countries. However, the Global Address Cleanse transform's engines may not provide address correction for all of those countries.

Related Information

To process U.S. territory with the USA engine [page 611]
Reference Guide: Country ISO codes and assignment engines [page 1395]

2.16.5.9.1 To change the default output country name

When you use the USA engine to process addresses from American Samoa, Guam, Northern Mariana Islands, Palau, Puerto Rico, and the U.S. Virgin Islands, the output region is AS, GU, MP, PW, PR, or VI, respectively. The output country, however, is the United States (US).

If you do not want the output country as the United States when processing addresses with the USA engine, set the *Use Postal Country Name* option to *No*.

These steps show you how to set the Use Postal Country Name in the Global Address Cleanse transform.

1. Open the *Global Address Cleanse* transform.
2. On the *Options* tab, expand ▶ *Standardization Options* ▶ *Country* ▶ *Options* ◀.
3. For the *Use Postal Country Name* option, select *No*.

Related Information

Supported countries (Global Address Cleanse) [page 611]

2.16.5.9.2 Set a default country

> i Note
>
> Run Country ID processing only if you are:

- Using two or more of the engines and your input addresses contain country data (such as the two-character ISO code or a country name).
- Using only one engine, but your input data contains addresses from multiple countries.

1. Open the *Global Address Cleanse* transform.
2. On the *Options* tab, expand *Country ID Options*, and then for the *Country ID Mode* option select *Assign*.
   This value directs the transform to use *Country ID* to assign the country. If *Country ID* cannot assign the country, it will use the value in *Country Name*.
3. For the *Country Name* option, select the country that you want to use as a default country.

   The transform will use this country only when *Country ID* cannot assign a country. If you do not want a default country, select *None*.
4. For the *Script Code* option, select the type of script code that represents your data.

   The *LATN* option provides script code for most types of data. However, if you are processing Japanese data, select *KANA*

Related Information

Identify the country of destination [page 593]

To set a constant country [page 612]

2.16.5.9.3 To set a constant country

1. Open the *Global Address Cleanse* transform.
2. On the *Options* tab, expand *Country ID Options*, and then for the *Country ID Mode* option select *Constant*.
   This value tells the transform to take the country information from the *Country Name* and *Script Code* options (instead of running *Country ID* processing).
3. For the *Country Name* option, select the country that represents all your input data.
4. For the *Script Code* option, select the type of script code that represents your data.

   The *LATN* option provides script code for most types of data. However, if you are processing Japanese data, select *KANA*

Related Information

Identify the country of destination [page 593]

Set a default country [page 611]

2.16.5.10 New Zealand certification

New Zealand certification enables you to process New Zealand addresses and qualify for mailing discounts with the New Zealand Post.

2.16.5.10.1 To enable New Zealand certification

You need to purchase the New Zealand directory data and obtain a customer number from the New Zealand Post before you can use the New Zealand certification option.

To process New Zealand addresses that qualify for mailing discounts:

1. In the Global Address Cleanse Transform, enable ▶ *Report and Analysis* ▸ *Generate Report Data* ◀.
2. In the Global Address Cleanse Transform, set ▶ *Country Options* ▸ *Disable Certification* ◀ to *No*.

   > Note
   >
   > The software does not produce the New Zealand Statement of Accuracy (SOA) report when this option is set to *Yes*.

3. In the Global Address Transform, complete all applicable options in the ▶ *Global Address* ▸ *Report Options* ▸ *New Zealand* ◀ subgroup.
4. In the Global Address Cleanse Transform, set ▶ *Engines* ▸ *Global Address* ◀ to *Yes*.

After you run the job and produce the New Zealand Statement of Accuracy (SOA) report, you need to rename the New Zealand Statement of Accuracy (SOA) report and New Zealand Statement of Accuracy (SOA) Production Log before submitting your mailing. For more information on the required naming format, See *New Zealand SOA Report and SOA production log file* [page 613].

Related Information

Management Console Guide: New Zealand Statement of Accuracy (SOA) report [page 2012]
Reference Guide: Report options for New Zealand [page 1217]

2.16.5.10.2 New Zealand SOA Report and SOA production log file

New Zealand Statement of Accuracy (SOA) Report

The New Zealand Statement of Accuracy (SOA) report includes statistical information about address cleansing for New Zealand.

New Zealand Statement of Accuracy (SOA) Production Log

The New Zealand Statement of Accuracy (SOA) production log contains identical information as the SOA report in a pipe-delimited ASCII text file (with a header record). The software creates the SOA production log by extracting data from the Sendrightaddraccuracy table within the repository. The software appends a new record to the Sendrightaddraccuracy table each time a file is processed with the DISABLE_CERTIFICATION option set to *No*. If the DISABLE_CERTIFICATION option is set to *Yes*, the software does not produce the SOA report and an entry will not be appended to the Sendrightaddraccuracy table. Mailers must retain the production log file for at least 2 years.

The default location of the SOA production log is `<INSTALL_DIR>\Data Services\DataQuality\certifications\CertificationLogs`.

Mailing requirements

The SOA report and production log are only required when you submit the data processed for a mailing and want to receive postage discounts. Submit the SOA production log at least once a month. Submit an SOA report for each file that is processed for mailing discounts.

File naming format

The SOA production log and SOA report must have a file name in the following format:

Production Log -

```
[SOA% (9999)]_[SOA Expiry Date (YYYYMMDD)]_[SOA ID].txt
```

SOA Report -

```
[SOA% (9999)]_[SOA Expiry Date (YYYYMMDD)]_[SOA ID].PDF
```

Example

An SOA with:

SOA % = 94.3%

SOA expiry date = 15 Oct 2008

SOA ID = AGD07_12345678

The file names will be:

Production Log - `0943_20081015_AGD07_12345678.txt`

SOA Report - `0943_20081015_AGD07_12345678.pdf`

Related Information

Management Console Guide: New Zealand Statement of Accuracy (SOA) report [page 2012]
Management Console Guide: Exporting New Zealand SOA certification logs [page 1879]
Reference Guide: Report options for New Zealand [page 1217]

2.16.5.10.3 The New Zealand certification blueprint

Do the following to edit the blueprint, run the job for New Zealand certification, and generate the SOA production log file:

1. Import `nz_sendright_certification.atl` located in the DataQuality\certifications folder in the location where you installed the software. The default location is **<INSTALL_DIR>**`\Data Services\DataQuality\certifications`.
   The import adds the following objects to the repository:
   - The project DataQualityCertifications
   - The job Job_DqBatchNewZealand_SOAProductionLog
   - The data flow DF_DqBatchNewZealand_SOAProductionLog
   - The datastore DataQualityCertifications
   - The file format DqNewZealandSOAProductionLog
2. Edit the datastore DataQualityCertifications. Follow the steps listed in *Editing the datastore* [page 616].
3. Optional: By default, the software places the SOA Production Log in **<INSTALL_DIR>**`\Data Services\DataQuality\certifications\CertificationLogs`. If the default location is acceptable, ignore this step. If you want to output the production log file to a different location, edit the substitution parameter configuration. From the Designer, access *Tools* › *Substitution Parameter Configurations* and change the path location in Configuration1 for the substitution parameter $$CertificationLogPath to the location of your choice.
4. Run the job Job_DqBatchNewZealand_SOAProductionLog.
   The job produces an SOA Production Log called `SOAPerc_SOAExpDate_SOAId.txt` in the default location or the location you specified in the substitution parameter configuration.
5. Rename the `SOAPerc_SOAExpDate_SOAId.txt` file using data in the last record in the log file and the file naming format described in *New Zealand SOA Report and SOA production log file* [page 613].

Related Information

New Zealand SOA Report and SOA production log file [page 613]
Management Console Guide: New Zealand Statement of Accuracy (SOA) report [page 2012]

2.16.5.10.4 Editing the datastore

After you download the blueprint `.zip` file to the appropriate folder, unzip it, and import the .atl file in the software, you must edit the DataQualityCertifications datastore.

To edit the datastore:

1. Select the *Datastores* tab of the Local Object Library, right-click DataQualityCertifications and select *Edit*.
2. Click *Advanced* to expand the Edit Datastore DataQualityCertifications window.

   > Note
   >
   > Skip step 3 if you have Microsoft SQL Server 2000 or 2005 as a datastore database type.

3. Click *Edit*.
4. Find the column for your database type, change *Default configuration* to **`Yes`**, and click *OK*.

   > Note
   >
   > If you are using a version of Oracle other than Oracle 9i, perform the following substeps:

   a) In the toolbar, click *Create New Configuration*.
   b) Enter your information, including the Oracle database version that you are using, and then click *OK*.
   c) Click *Close* on the Added New Values - Modified Objects window.
   d) In the new column that appears to the right of the previous columns, select *Yes* for the *Default configuration*.
   e) Enter your information for the *Database connection name*, *User name*, and *Password* options.
   f) In *DBO*, enter your schema name.
   g) In *Code Page*, select *cp1252* and then click *OK*.
5. At the Edit Datastore DataQualityCertifications window, enter your repository connection information in place of the CHANGE_THIS values. (You may have to change three or four options, depending on your repository type.)
6. Expand the *Aliases* group and enter your owner name in place of the CHANGE_THIS value. If you are using Microsoft SQL Server, set this value to **`DBO`**.
7. Click *OK*.
   If the window closes without any error message, then the database is successfully connected.

2.16.5.11 Global Address Cleanse Suggestion List option

The Global Address Cleanse transform's Suggestion List processing option is used in transactional projects to complete and populate addresses that have minimal data. Suggestion lists can offer suggestions for possible matches if an exact match is not found.

This option is beneficial in situations where you want to extract addresses not completely assigned by an automated process, and run through the system to find a list of possible matches. Based on the given input address, the Global Address Cleanse transform performs an error-tolerant search in the address directory and returns a list of possible matches. From the suggestion list returned, you can select the correct suggestion and update the database accordingly.

ℹ Note

No certification with suggestion lists: If you use the Canada engine or Global Address engine for Australia and New Zealand, you cannot certify your mailing for SERP, AMAS, or New Zealand certification.

Start with a sample transform

If you want to use the suggestion lists feature, it is best to start with the sample transforms that is configured for it. The sample transform, GlobalSuggestions_AddressCleanse is configured to cleanse Latin-1 address data in any supported country using the Suggestion List feature.

Related Information

Extracting data quality XML strings using extract_from_xml function [page 370]
Reference Guide: Suggestion List options [page 1215]

2.16.5.12 Global Suggestion List transform

The Global Suggestion List transform allows you to query addresses with minimal data (allowing the use of wildcards), and it can offer a list of suggestions for possible matches.

It is a beneficial tool for a call center environment, where operators need to enter minimum input (number of keystrokes) to find the caller's delivery address. For example, if the operator is on the phone with a caller from the United Kingdom, the application prompts for the postcode and address range. Global Suggestion List is used to look up the address with quick-entry.

The Global Suggestion List transform requires the two-character ISO country code on input. Therefore, you may want to place a transform, such as the Country ID transform, that outputs the ISO_Country_Code_2Char field before the Global Suggestion Lists transform. The Global Suggestion List transform is available for use with the Canada, Global Address, and USA engines.

ℹ Note

No certification with suggestion lists: If you use the Canada, Global Address, or USA engines for Australia and New Zealand, you cannot certify your mailing for SERP, CASS, AMAS, or New Zealand certification.

ℹ Note

This option does not support processing of Japanese or Chinese address data.

Start with a sample transform

If you want to use the Global Suggestion List transform, it is best to start with one of the sample transforms that is configured for it. The following sample tranforms are available.

Sample transform	Description
GlobalSuggestions	A sample transform configured to generate a suggestion list for Latin-1 address data in any supported country.
UKSuggestions	A sample transform configured to generate a suggestion list for partial address data in the United Kingdom.

2.16.6 Beyond the basic address cleansing

The USA Regulatory Address Cleanse transform offers many additional address cleanse features for U.S. addresses. These features extend address cleansing beyond the basic parsing and standardizing. To read about the USA Regulatory Address Cleanse transform and its options, see the *Reference Guide*.

2.16.6.1 USPS DPV®

Delivery Point Validation® is a USPS product developed to assist users in validating the accuracy of their address information. DPV compares Postcode2 information against the DPV directories to identify known addresses and potential problems with the address that may cause an address to become undeliverable.

DPV is available for U.S. data in the USA Regulatory Address Cleanse transform only.

> **Note**
>
> DPV processing is required for CASS certification. If you are not processing for CASS certification, you can choose to run your jobs in non-certified mode and still enable DPV.

> **Caution**
>
> If you choose to disable DPV processing, the software will not generate the CASS-required documentation and your mailing won't be eligible for postal discounts.

Related Information

To enable DPV [page 624]

Run DPV in non certified mode [page 625]

2.16.6.1.1 Benefits of DPV

DPV can be beneficial in the following areas:

- Mailing: DPV helps to screen out undeliverable-as-addressed (UAA) mail and helps to reduce mailing costs.
- Information quality: DPV increases the level of data accuracy through the ability to verify an address down to the individual house, suite, or apartment instead of only block face.
- Increased assignment rate: DPV may increase assignment rate through the use of DPV tiebreaking to resolve a tie when other tie-breaking methods are not conclusive.
- Preventing mail-order-fraud: DPV can eliminate shipping of merchandise to individuals who place fraudulent orders by verifying valid delivery addresses and Commercial Mail Receiving Agencies (CMRA).

2.16.6.1.2 DPV security

The USPS has instituted processes that monitor the use of DPV. Each company that purchases the DPV functionality is required to sign a legal agreement stating that it will not attempt to misuse the DPV product. If a user abuses the DPV product, the USPS has the right to prohibit the user from using DPV in the future.

2.16.6.1.2.1 DPV false positive addresses

The USPS has included false positive addresses in the DPV directories as an added security to prevent DPV abuse. Depending on what type of user you are and your license key codes, the software's behavior varies when it encounters a false positive address. The following table explains the behaviors for each user type:

User type	Software behavior	Read about:
End users	DPV processing is terminated.	Obtaining DPV unlock code from SAP Support
End users with a stop processing alternative agreement	DPV processing continues.	Sending false positive logs to the USPS
Service providers	DPV processing continues.	Sending false positive logs to the USPS

Related Information

Stop Processing Alternative [page 620]
Obtaining DPV unlock code from SAP [page 622]
Sending DPV false positive logs to the USPS [page 622]

2.16.6.1.2.2 Stop Processing Alternative

End users may establish a Stop Processing Alternative agreement with the USPS and SAP.

Establishing a stop processing agreement allows you to bypass any future directory locks. The Stop Processing Alternative is not an option in the software, it is a key code that you obtain from SAP Support.

First you must obtain the proper permissions from the USPS and then provide proof of permission to SAP Support. Support will then provide a key code that disables the directory locking function in the software.

➜ Remember

When you obtain the Stop Processing Alternative key code from SAP Support, enter it into the SAP License Manager. With the Stop Processing Alternative key code in place, the software takes the following actions when a false positive is encountered:

- Marks the record as a false positive.
- Generates a log file containing the false positive address.
- Notes the path to the log files in the error log.
- Generates a US Regulatory Locking Report containing the path to the log file.
- Continues processing your job.

Even though your job continues processing, you are required to send the false positive log file to the USPS to notify them that a false positive address was detected. The USPS must release the list before you can use it for processing.

Related Information

Sending DPV false positive logs to the USPS [page 622]

2.16.6.1.2.3 DPV false positive logs

The software generates a false positive log file any time it encounters a false positive record, regardless of how your job is set up. The software creates a separate log file for each mailing list that contains a false positive. If multiple false positives exist within one mailing list, the software writes them all to the same log file.

2.16.6.1.2.3.1 DPV log file name and location

The software stores DPV log files in the directory specified in the *USPS Log Path* option in the Reference Files group.

ℹ Note

The USPS log path that you enter must be writable. An error is issued if you have entered a path that is not writable.

Log file naming convention

The software automatically names DPV false positive logs with the following format: `dpvl####.log`

The #### portion of the naming format is a number between 0001 and 9999. For example, the first log file generated is `dpvl0001.log`, the next one is `dpvl0002.log`, and so on.

ℹ Note

When you have set the data flow degree of parallelism to greater than 1, or you have enabled the run as a separate process option, the software generates one log per thread or process. During a job run, if the software encounters only one false positive record, one log will be generated. However, if it encounters more than one false positive record and the records are processed on different threads or processes, then the software will generate one log for each thread that processes a false positive record.

Related Information

Performance Optimization Guide: Using parallel execution [page 2137]

2.16.6.1.2.4 DPV locking for end users

This locking behavior is applicable for end users or users who are DSF2 licensees that have DSF2 disabled in the job

When the software finds a false positive address, DPV processing is discontinued for the remainder of the data flow. The software also takes the following actions:

- Marks the record as a false positive address.
- Issues a message in the error log stating that a DPV false positive address was encountered.
- Includes the false positive address and lock code in the error log.
- Continues processing your data flow without DPV processing.
- Generates a lock code.
- Generates a false positive log.
- Generates a US Regulatory Locking Report that contains the false positive address and the lock code. (Report generation must be enabled in the USA Regulatory Address Cleanse transform.)

To restore DPV functionality, users must obtain a DPV unlock code from SAP Support.

Related Information

Obtaining LACSLink unlock code from SAP Support [page 622]

2.16.6.1.2.5 Obtaining DPV unlock code from SAP

These steps are applicable for end users who do not have a Stop Processing Alternative agreement with the USPS. When you receive a processing message that DPV false positive addresses are present in your address list, use the SAP USPS Unlock Utility to obtain an unlock code.

1. Navigate to *https://websmp205.sap-ag.de/bosap-unlock* to open the SAP Service Market Place (SMP) unlock utility page.
2. Click *Retrieve USPS Unlock Code*.
3. Click *Search* and select an applicable Data Services system from the list.
4. Enter the lock code found in the `dpvx.txt` file (location is specified in the *DPV Path* option in the Reference Files group).
5. Select DPV as the lock type.
6. Select BOJ-EIM-DS as the component.
7. Enter the locking address that is listed in the `dpvx.txt` file.
8. Attach the `dpvl####.log` file (location is specified in the *USPS Log Path* option in the Reference Files group).
9. Click *Submit*.
   The unlock code displays.
10. Copy the unlock code and paste it into the `dpvw.txt` file, replacing all contents of the file with the unlock code (location is specified in the *DPV path* option of the Reference Files group).
11. Remove the record that caused the lock from the database, and delete the `dpvl####.log` file before processing the list again.

    Tip

    Keep in mind that you can only use the unlock code once. If the software detects another false-positive (even if it is the same record), you must retrieve a new LACSLink unlock code.

If an unlock code could not be generated, a message is still created and is processed by a Technical Customer Assurance engineer (during regular business hours).

> Note
>
> If you are an end user who has a Stop Processing Alternative agreement, follow the steps to send the false positive log to the USPS.

2.16.6.1.2.6 Sending DPV false positive logs to the USPS

Service providers should follow these steps after receiving a processing message that DPV false positive addresses are present in their address list. End users with a Stop Processing Alternative agreement should follow these steps after receiving a processing message that DPV false positive addresses are present in their address list.

1. Send an email to the USPS NCSC at Dsf2stop@usps.gov, and include the following information:
   - Type “DPV False Positive” as the subject line
   - Attach the `dpvl####.log` file or files that were generated by the software (location is specified in the *USPS Log Path* directory option in the Reference Files group)

   The USPS NCSC uses the information to determine whether the list can be returned to the mailer.
2. After the USPS NCSC has released the list that contained the locked or false positive record:
   - Delete the corresponding log file or files
   - Remove the record that caused the lock from the list and reprocess the file

If you are an end user who does not have a Stop Processing Alternative agreement, follow the steps to retrieve the DPV unlock code from SAP Support.

Related Information

Obtaining DPV lock code from SAP BusinessObjects Support [page 622]

2.16.6.1.3 DPV monthly directories

DPV directories are shipped monthly with the USPS directories in accordance with USPS guidelines.

The directories expire in 105 days. The date on the DPV directories must be the same date as the Address directory.

Do not rename any of the files. DPV will not run if the file names are changed. Here is a list of the DPV directories:

- `dpva.dir`
- `dpvb.dir`
- `dpvc.dir`
- `dpvd.dir`
- `dpv_vacant.dir`
- `dpv_no_stats.dir`

2.16.6.1.4 Required information in the job setup

When you set up for DPV processing, the following options in the USPS License Information group are required:

- *Customer Company Name*
- *Customer Company Address*
- *Customer Company Locality*
- *Customer Company Region*

- *Customer Company Postcode1*
- *Customer Company Postcode2*

2.16.6.1.5 To enable DPV

> i Note
>
> DPV is required for CASS.

In addition to the required customer company information that you enter into the USPS License Information group, set the following options to perform DPV processing:

1. Open the USA Regulatory Address Cleanse transform.
2. Open the *Options* tab. Expand the Assignment Options group, and select Yes for the *Enable DPV* option.
3. In the Reference Files group, enter the path for your DPV directories in the *DPV Path* option.

   > i Note
   >
   > DPV can run only when the location for all the DPV directories have been entered and none of the DPV directory files have been renamed.

4. Set a directory for the DPV log file in the *USPS Path* option. Use the substitution variable $ $CertificationLogPath if you have it set up.
5. In the Report and Analysis group, select Yes for the *Generate Report Data* option.

2.16.6.1.6 DPV output fields

Several output fields are available for reporting DPV processing results:

- DPV_CMRA
- DPV_Footnote
- DPV_NoStats
- DPV_Status
- DPV_Vacant

For full descriptions of these output fields, refer to the *Reference Guide* or view the Data Services Help information that appears when you open the Output tab of the USA Regulatory Address Cleanse transform.

Related Information

Reference Guide: Data Quality fields, USA Regulatory Address Cleanse fields, Output fields [page 1353]

2.16.6.1.7 Non certified mode

You can set up your jobs with DPV disabled if you are not a CASS customer but you want a Postcode2 added to your addresses. The non-CASS option, *Assign Postcode2 Not DPV Validated*, enables the software to assign a Postcode2 when an address does not DPV-confirm.

> ⚠ Caution
>
> If you choose to disable DPV processing, the software does not generate the CASS-required documentation and your mailing won't be eligible for postal discounts.

2.16.6.1.7.1 Enable Non-Certified mode

To run your job in non certified mode, follow these setup steps:

1. In the Assignment Options group, set the *Enable DPV* option to No.
2. In the Non Certified options group, set the *Disable Certification* to Yes.
3. In the Non Certified options group, set the *Assign Postcode2 Not DPV Validated* to Yes.

> ⚠ Caution
>
> The software blanks out all Postcode2 information in your data if you disable DPV processing and you disable the *Assign Postcode2 Not DPV Validated* option. This includes Postcode2 information provided in your input file.

2.16.6.1.8 DPV performance

Due to additional time required to perform DPV processing, you may see a change in processing time. Processing time may vary with the DPV feature based on operating system, system configuration, and other variables that may be unique to your operating environment.

You can decrease the time required for DPV processing by loading DPV directories into system memory before processing.

2.16.6.1.8.1 Memory usage

You may need to install additional memory on your operating system for DPV processing. We recommend a minimum of 768 MB to process with DPV enabled.

To determine the amount of memory required to run with DPV enabled, check the size of the DPV directories (recently about 600 MB[1]) and add that to the amount of memory required to run the software.

The size of the DPV directories will vary depending on the amount of new data in each directory release.

Make sure that your computer has enough memory available before performing DPV processing.

To find the amount of disk space required to cache the directories, see the *Supported Platforms* document in the SAP Support portal. Find link information in the SAP Information resources table (see the link below).

Related Information

SAP Business Objects information resources, Supported platforms (Product Availability Matrix) [page 169]

2.16.6.1.8.2 Cache DPV directories

To better manage memory usage when you have enabled DPV processing, choose to cache the DPV directories.

2.16.6.1.8.3 To cach DPV directories

To set up your job for DPV caching, follow these steps:

1. In the Transform Performance group, set the *Cache DPV Directories* option to Yes.
2. In the same group, set the *Insufficient Cache Memory Action* to one of the following:

Option	Description
Error	Software issues an error and terminates the transform.
Continue	Software attempts to continue initialization without caching.

2.16.6.1.8.4 Running multiple jobs with DPV

When running multiple DPV jobs and loading directories into memory, you should add a 10-second pause between jobs to allow time for the memory to be released. For more information about setting this properly, see your operating system manual.

If you don't add a 10-second pause between jobs, there may not be enough time for your system to release the memory used for caching the directories from the first job. The next job waiting to process may error out or access the directories from disk if there is not enough memory to cache directories. This may result in performance degradation.

[1] The directory size is subject to change each time new DPV directories are installed.

2.16.6.1.9 DPV information in US Addressing Report

The US Addressing Report automatically generates when you have enabled reporting in your job. The following sections of the US Addressing Report contain DPV information:

- DPV Return Codes
- Delivery Point Validation (DPV) Summary

For information about the US Addressing Report, or other Data Quality reports, see the *Management Console Guide*.

Related Information

Management Console: Data Quality reports, US Addressing Report [page 2004]

2.16.6.1.10 DPV No Stats indicators

The USPS uses No Stats indicators to mark addresses that fall under the No Stats category. The software uses the No Stats table when you have DPV or DSF2 turned on in your job. The USPS puts No Stats addresses in three categories:

- Addresses that do not have delivery established yet.
- Addresses that receive mail as part of a drop.
- Addresses that have been vacant for a certain period of time.

2.16.6.1.10.1 No Stats table

You must install the No Stats table (`dpv_no_stats.dir`) before the software performs DPV or DSF2 processing. The No Stats table is supplied by SAP BusinessObjects with the DPV directory install.

The software automatically checks for the No Stats table in the directory folder that you indicate in your job setup. The software performs DPV and DSF2 processing based on the install status of the directory.

dpv_no_stats.dir	Results
Installed	The software automatically outputs No Stats indicators when you include the DPV_NoStats output field in your job.
Not installed	The software automatically skips the No Stats processing and does not issue an error message. The software will perform DPV processing but won't populate the DPV_NoStat output field.

2.16.6.1.10.2 No Stats output field

Use the DPV_NoStats output field to post No Stat indicator information to an output file.

No Stat means that the address is a vacant property, it receives mail as a part of a drop, or it does not have an established delivery yet.

Related Information

DPV output fields [page 624]

2.16.6.1.11 DPV Vacant indicators

The software provides vacant information in output fields and reports using DPV vacant counts. The USPS DPV vacant lookup table is supplied by SAP BusinessObjects with the DPV directory install.

The USPS uses DPV vacant indicators to mark addresses that fall under the vacant category. The software uses DPV vacant indicators when you have DPV or DSF2 enabled in your job.

> ➜ Tip
>
> The USPS defines vacant as any delivery point that was active in the past, but is currently not occupied (usually over 90 days) and is not currently receiving mail delivery. The address could receive delivery again in the future. "Vacant" does not apply to seasonal addresses.

2.16.6.1.11.1 DPV address-attribute output field

Vacant indicators for the assigned address are available in the DPV_Vacant output field.

> i Note
>
> The US Addressing Report contains DPV Vacant counts in the DPV Summary section.

Related Information

DPV output fields [page 624]

Management Console: Data Quality reports, US Addressing Report [page 2003]

2.16.6.2 LACSLink®

LACSLink is a USPS product that is available for U.S. records with the USA Regulatory Address Cleanse transform only. LACSLink processing is required for CASS certification.

LACSLink updates addresses when the physical address does not move but the address has changed. For example, when the municipality changes rural route addresses to street-name addresses. Rural route conversions make it easier for police, fire, ambulance, and postal personnel to locate a rural address. LACSLink also converts addresses when streets are renamed or post office boxes renumbered.

LACSLink technology ensures that the data remains private and secure, and at the same time gives you easy access to the data. LACSLink is an integrated part of address processing; it is not an extra step. To obtain the new addresses, you must already have the old address data.

Related Information

How LACSLink works [page 632]
To control memory usage for LACSLink processing [page 636]
To disable LACSLink [page 635]
LACSLink security [page 629]

2.16.6.2.1 Benefits of LACSLink

LACSLink processing is required for all CASS customers.

If you process your data without LACSLink enabled, you won't get the CASS-required reports or postal discounts.

2.16.6.2.2 LACSLink security

The USPS has instituted processes that monitor the use of LACSLink. Each company that purchases the LACSLink functionality is required to sign a legal agreement stating that it will not attempt to misuse the LACSLink product. If a user abuses the LACSLink product, the USPS has the right to prohibit the user from using LACSLink in the future.

2.16.6.2.2.1 LACSLink false positive addresses

The USPS has included false positive addresses in the LACSLink directories as an added security to prevent LACSLink abuse. Depending on what type of user you are and your license key codes, the software's behavior varies when it encounters a false positive address. The following table explains the behaviors for each user type:

User type	Software behavior	Read about:
End users	LACSLink processing is terminated.	Obtaining the LACSLink unlock code from SAP Support
End users with a Stop Processing Alternative agreement	LACSLink processing continues.	Sending false positive logs to the USPS
Service providers	LACSLink processing continues.	Sending false positive logs to the USPS

Related Information

Alternate stop processing agreements [page 620]
Obtaining LACSLink unlock code from SAP [page 631]
Sending LACSLink false positive logs to the USPS [page 632]

2.16.6.2.2.2 LACSLink false positive logs

The software generates a false-positive log file any time it encounters a false positive record, regardless of how your job is set up. The software creates a separate log file for each mailing list that contains a false positive. If multiple false positives exist within one mailing list, the software writes them all to the same log file.

2.16.6.2.2.2.1 LACSLink log file location

The software stores LACSLink log files in the directory specified for the *USPS Log Path* in the Reference Files group.

> i Note
>
> The USPS log path that you enter must be writable. An error is issued if you have entered a path that is not writable.

The software names LACSLink false positive logs `lacsl`**`<###>`**`.log`, where **`<###>`** is a number between 001 and 999. For example, the first log file generated is `lacsl001.log`, the next one is `lacsl002.log`, and so on.

> i Note
>
> When you have set the data flow degree of parallelism to greater than 1, the software generates one log per thread. During a job run, if the software encounters only one false positive record, one log will be generated. However, if it encounters more than one false positive record and the records are processed on different threads, then the software will generate one log for each thread that processes a false positive record.

Related Information

Performance Optimization Guide: Using parallel execution [page 2137]

2.16.6.2.2.3 LACSLink locking for end users

This locking behavior is applicable for end users or users who are DSF2 licensees that have DSF2 disabled in the job.

When the software finds a false positive address, LACSLink processing is discontinued for the remainder of the job processing. The software takes the following actions:

- Marks the record as a false positive address.
- Issues a message in the error log that a LACSLink false positive address was encountered.
- Includes the false positive address and lock code in the error log.
- Continues processing your data flow without LACSLink processing.
- Generates a lock code.
- Generates a false positive error log.
- Generates a US Regulatory Locking Report that contains the false positive address and the lock code (Report generation must be enabled in the USA Regulatory Address Cleanse transform.

To restore LACSLink functionality, users must obtain a LACSLink unlock code from SAP Support.

2.16.6.2.2.4 Obtaining LACSLink unlock code from SAP

These steps are applicable for end users who do not have a Stop Processing Alternative agreement with the USPS. When you receive a processing message that LACSLink false positive addresses are present in your address list, use the SAP USPS Unlock Utility to obtain an unlock code.

1. Navigate to *https://websmp205.sap-ag.de/bosap-unlock* to open the SAP Service Market Place (SMP) unlock utility page.
2. Click *Retrieve USPS Unlock Code*.
3. Click *Search* and select an applicable Data Services system from the list.
4. Enter the lock code found in the `lacsx.txt` file (location is specified in the *LACSLink Path* option in the Reference Files group).
5. Select LACSLink as the lock type.
6. Select BOJ-EIM-DS as the component.
7. Enter the locking address that is listed in the `lacsx.txt` file.
8. Attach the `lacsl####.log` file (location specified in the *USPS Log Path* option in the Reference Files group).
9. Click *Submit*.
   The unlock code displays.
10. Copy the unlock code and paste it into the `lacsw.txt` file, replacing all contents of the file with the unlock code (location is specified in the *LACSLink path* option in the Reference Files group).

11. Remove the record that caused the lock from the database, and delete the lacsl####.log file before processing the list again.

   > ➜ Tip
   >
   > Keep in mind that you can only use the unlock code once. If the software detects another false-positive (even if it is the same record), you must retrieve a new LACSLink unlock code.

If an unlock code could not be generated, a message is still created and is processed by a Technical Customer Assurance engineer (during regular business hours).

> i Note
>
> If you are an end user who has a Stop Processing Alternative agreement, follow the steps to send the false positive log to the USPS.

2.16.6.2.2.5 Sending LACSLink false positive logs to the USPS

Service providers should follow these steps after receiving a processing message that LACSLink false positive addresses are present in their address list. End users with a Stop Processing Alternative agreement should follow these steps after receiving a processing message that LACSLink false positive addresses are present in their address list.

1. Send an email to the USPS at Dsf2stop@usps.gov. Include the following:
   ○ Type “LACSLink False Positive” as the subject line
   ○ Attach the `lacsl###.log` file or files that were generated by the software (location specified in the *USPS Log Files* option in the Reference Files group).

   The USPS NCSC uses the information to determine whether or not the list can be returned to the mailer.
2. After the USPS NCSC has released the list that contained the locked or false positive record:
   ○ Delete the corresponding log file or files
   ○ Remove the record that caused the lock from the list and reprocess the file

If you are an end user who does not have a Stop Processing Alternative agreement, follow the steps to retrieve the LACSLink unlock code from SAP Support.

Related Information

Obtaining LACSLink unlock code from SAP [page 631]

2.16.6.2.3 How LACSLink works

LACSLink provides a new address when one is available. LACSLink follows these steps when processing an address:

1. The USA Regulatory Address Cleanse transform standardizes the input address.
2. The transform looks for a matching address in the LACSLink data.
3. If a match is found, the transform outputs the LACSLink-converted address and other LACSLink information.

Related Information

To control memory usage for LACSLink processing [page 636]
LACSLink (USA Regulatory Address Cleanse) [page 629]

2.16.6.2.4 Conditions for address processing

The transform does not process all of your addresses with LACSLink when it is enabled. Here are the conditions under which your data is passed into LACSLink processing:

- The address is found in the address directory, and it is flagged as a LACS-convertible record within the address directory.
- The address is found in the address directory, and, even though a rural route or highway contract default assignment was made, the record wasn't flagged as LACS convertible.
- The address is not found in the address directory, but the record contains enough information to be sent into LACSLink.

For example, the following table shows an address that was found in the address directory as a LACS-convertible address.

Original address	After LACSLink conversion
RR2 BOX 204 DU BOIS PA 15801	463 SHOWERS RD DU BOIS PA 15801-66675

2.16.6.2.5 Sample transform configuration

LACSLink processing is enabled by default in the sample transform configuration because it is required for CASS certification. The sample transform configuration is named USARegulatory_AddressCleanse and is found under the USA_Regulatory_Address_Cleanse group in the Object Library.

2.16.6.2.6 LACSLink directory files

SAP ships the LACSLink directory files with the U.S. National Directory update. The LACSLink directory files require about 600 MB of additional hard drive space. The LACSLink directories include the following:

- `lacsw.txt`
- `lacsx.txt`
- `lacsy.ll`
- `lacsz.ll`

⚠ Caution

The LACSLink directories must reside on the hard drive in the same directory as the LACSLink supporting files. Do not rename any of the files. LACSLink will not run if the file names are changed.

2.16.6.2.6.1 Directory expiration and updates

LACSLink directories expire in 105 days. LACSLink directories must have the same date as the other directories that you are using from the U.S. National Directories.

2.16.6.2.7 To enable LACSLink

LACSLink is enabled by default in the USA Regulatory Address Cleanse transform. If you need to re-enable the option, follow these steps:

1. Open the USA Regulatory Address Cleanse transform and open the *Options* tab.
2. Expand the Processing Options group
3. select Yes in the *Enable LACSLink* option.
4. Enter the LACSLink path for the *LACSLink Path* option In the Reference Files group. You can use the substitution variable $$RefFilesAddressCleanse if you have it set up.
5. Complete the required fields in the USPS License Information group.

2.16.6.2.7.1 Required information in the job setup

All users running LACSLink must include required information in the USPS License Information group. The required options include the following:

- Customer Company Name
- Customer Company Address
- Customer Company Locality
- Customer Company Region
- Customer Company Postcode1
- Customer Company Postcode2
- Customer Company Phone

2.16.6.2.7.2 To disable LACSLink

LACSLink is enabled by default in the USA Regulatory Address Cleanse transform configuration because it is required for CASS processing. Therefore, you must disable CASS certification in order to disable LACSLink.

1. In the USA Regulatory Address Cleanse transform configuration, open the *Options* tab.
2. Open the Non Certified Options group.
3. Select Yes for the *Disable Certification* option.
4. Open the Assignment Option group.
5. Select No for the *Enable LACSLink* option.

Related Information

LACSLink® [page 629]

2.16.6.2.7.3 Reasons for errors

If your job setup is missing information in the USPS License Information group, and you have DPV and/or LACSLink enabled in your job, you will get error messages based on these specific situations:

Reason for error	Description
Missing required options	When your job setup does not include the required parameters in the USPS License Information group, and you have DPV and/or LACSLink enabled in your job, the software issues a verification error.
Unwritable Log File directory	If the path that you specified for the *USPS Log Path* option in the Reference Files group is not writable, the software issues an error.

2.16.6.2.8 LACSLink output fields

Several output fields are available for reporting LACSLink processing results.

You must enable LACSLink, and include these output fields in your job setup, before the software posts information to these fields.

Field name	Length	Description
LACSLINK_QUERY	50	Returns the pre-conversion address, populated only when LACSLink is enabled and a LACSLink lookup was attempted.

Field name	Length	Description
		This address will be in the standard USPS format (as shown in USPS Publication 28). However, when an address has both a unit designator and secondary unit, the unit designator is replaced by the character “#”. blank: No LACSLink lookup attempted.
LACSLINK_RETURN_CODE	2	Returns the match status for LACSLink processing: A = LACSLink record match. A converted address is provided in the address data fields. 00 = No match and no converted address. 09 = LACSLink matched an input address to an old address, which is a "high-rise default" address; no new address is provided. 14 = Found a LACSLink record, but couldn't convert the data to a deliverable address. 92 = LACSLink record matched after dropping the secondary number from input address. blank = No LACSLink lookup attempted.
LACSLINK_INDICATOR	1	Returns the conversion status of addresses processed by LACSLink. Y = Address converted by LACSLink (the LACSLink_Return_Code value is A). N = Address looked up with LACSLink but not converted. F = The address was a false-positive. S = LACSLink conversion was made, but it was necessary to drop the secondary information. blank: No LACSLink lookup attempted.

2.16.6.2.9 To control memory usage for LACSLink processing

The transform performance improves considerably if you cache the LACSLink directories. For the amount of disk space required to cache the directories, see the *Supported Platforms* document available in the *SAP Support* › *Documentation* › *Supported Platforms/PARs* section of the SAP Service Marketplace: *http://service.sap.com/bosap-support*.

If you do not have adequate system memory to load the LACSLink directories and the *Insufficient Cache Memory Action* is set to Error, a verification error message is displayed at run-time and the transform terminates. If the Continue option is chosen, the transform attempts to continue LACSLink processing without caching.

Open the *Options* tab of your USA Regulatory Address Cleanse transform configuration in your data flow. Follow these steps to load the LACSLink directories into your system memory:

1. Open the Transform Performance option group.
2. Select Yes for the *Cache LACSLink Directories* option.

Related Information

LACSLink (USA Regulatory Address Cleanse) [page 629]

2.16.6.2.10 LACSLink information in US Addressing Report

The US Addressing Report automatically generates when you have enabled reporting in your job. The following table lists the LACSLink sections in the US Addressing Report:

Section	Information
Locatable Address Conversion (LACSLink) Summary	Record counts and percentages for the following information: • LACSLink converted addresses • Addresses not LACSLink converted
LACSLink Return Codes	Record counts and percentages for the following information: • Converted • Secondary dropped • No match • Can't convert • High-rise default

2.16.6.2.11 USPS Form 3553

The USPS Form 3553 reports LACSLink counts. The LACS/LACSLink field shows the number of records that have a LACSLink Indicator of Y or S, if LACSLink processing is enabled. If LACSLink processing is not enabled, this field shows the number of LACS code count.

2.16.6.3 SuiteLink™

SuiteLink is an option in the USA Regulatory Address Cleanse transform.

SuiteLink uses a USPS directory that contains multiple files of specially indexed address information, such as secondary numbers and unit designators, for locations identified as high-rise default buildings.

With SuiteLink, you can build accurate and complete addresses by adding suite numbers to high-rise business addresses. With the secondary address information added to your addresses, more of your pieces are sorted by delivery sequence and delivered with accuracy and speed.

SuiteLink is required for CASS

SuiteLink is required when you process in CASS mode (and the *Disable certification* option is set to No). If you have disabled SuiteLink in your job setup, but you are in CASS mode, an error message is issued and processing does not continue.

2.16.6.3.1 Benefits of SuiteLink

Businesses that depend on website, mail, or in-store orders from customers will find that SuiteLink is a powerful money-saving tool. Also, businesses that have customers that reside in buildings which house several businesses will appreciate getting their marketing materials, bank statements, and orders delivered right to their door.

The addition of secondary number information to your addresses allows for the most efficient and cost-effective delivery sequencing and postage discounts.

Note

SuiteLink is required for those preparing CASS-compliant mailing lists.

2.16.6.3.2 How SuiteLink works

The software uses the data in the SuiteLink directories to add suite numbers to applicable addresses. The software matches a company name, a known high-rise address, and the CASS-certified postcode2 in your database to data in SuiteLink. When there is a match, the software creates a complete business address that includes the suite number.

Example

Assign suite number

This example shows a record that is processed through SuiteLink, and the output record with the assigned suite number.

The input record contains:

- Firm name (in FIRM input field)
- Known high-rise address
- CASS-certified postcode2

The SuiteLink directory contains:

- secondary numbers
- unit designators

The output record contains:

- the correct suite number

Input record	Output record
Telera 910 E Hamilton Ave Fl2 Campbell CA 95008 0610	TELERA 910 E HAMILTON AVE STE 200 CAMPBELL CA 95008 0625

2.16.6.3.3 SuiteLink directory

The SuiteLink directory is distributed monthly.

You must use SuiteLink directories with a `zip4us.dir` directory for the same month. (Enter the `zip4us.dir` path in the *Address Directory1* option of the Reference Files group in the USA Regulatory Address Cleanse transform.) For example, the December 2011 SuiteLink directory can be used with only the December 2011 `zip4us.dir` directory.

You cannot use a SuiteLink directory that is older than 60 days based on its release date. The software warns you 15 days before the directory expires. As with all directories, the software won't process your records with an expired SuiteLink directory.

2.16.6.3.4 To enable SuiteLink

SuiteLink is enabled by default in any of the sample transform configurations that are set up to be CASS-compliant (and the Disable certification option is set to No). For example, SuiteLink is enabled if you use the USA Regulatory_AddressCleanse sample transform configuration.

1. Open the USA Regulatory Address Cleanse transform in your data flow.
2. Open the *Options* tab.
3. Expand the Assignment Options group and set the *Enable SuiteLink* option to Yes.
4. In the Reference Files group, enter the SuiteLink directory path in the *SuiteLink Path* option. You can use the substitution variable $$RefFilesAddressCleanse if you have it set up with the directory location that contains your SuiteLink directories.
5. Optional: In the Transform Performance option group, set the *Cache SuiteLink Directories* option to Yes so that the SuiteLink directories are cached in memory.

   ℹ Note

   Ensure that you have sufficient RAM to cache the SuiteLink directories before you enable this option.

2.16.6.3.5 Improve processing speed

You may increase SuiteLink processing speed if you load the SuiteLink directories into memory. To activate this option, go to the Transform Performance group and set the *Cache SuiteLink Directories* to Yes.

2.16.6.3.6 SuiteLink return codes in US Addressing Report

SuiteLink return code information is available in the SuiteLink Return Codes section of the US Addressing Report.

The US Addressing Report shows the record count and percentage for the following return codes:

A = Secondary exists and assignment made

00 = Lookup was attempted but no assignment

2.16.6.4 USPS DSF2®

DSF2 is a USPS-licensed product that you can use to validate addresses, add delivery sequence information, and add DSF2 address attributes to addresses.

Two DSF2 features are supported in Data Services:

- DSF2 Augment in the USA Regulatory Address Cleanse transform
- DSF2 Walk Sequence in the DSF2 Walk Sequencer transform

> Note
>
> USPS DSF2 data is available only to USPS-certified DSF2 licensees.

Related Information

DSF2 Walk sequencing [page 645]

2.16.6.4.1 Validate addresses

DSF2 helps reduce the quantity of undeliverable-as-addressed (UAA) mail and keeps mailing costs down. DSF2 uses DPV® to validate addresses and identify inaccurate or incomplete addresses.

Related Information

DPV overview [page 618]

2.16.6.4.2 Add address attributes

DSF2 adds address attributes (information about the addresses) to your data. Use the attribute information to create more targeted mailings.

2.16.6.4.3 Add delivery sequence information

DSF2 adds delivery sequence information to your data, which you can use to qualify for walk-sequence discounts. This information is sometimes called walk sequencing or pseudo sequencing.

Related Information

Walk sequencing [page 645]
Pseudo sequencing [page 646]

2.16.6.4.4 Benefits of DSF2

Those who want to target their mail to specific types of addresses and those who want to earn additional postal discounts will appreciate what DSF2 can do.

The DSF2 address-attribute data provides mailers with knowledge about the address above and beyond what is necessary to accurately format the addresses. Address-attribute data allows mailers to produce more targeted mailings.

For example, If you plan to send out a coupon for your lawn-care service business, you do not want to send it to apartment dwellers (they may not have a lawn). You want your coupon to go to residential addresses that are not centralized in an apartment building.

With the DSF2 information you can walk-sequence your mailings to achieve the best possible postal discounts by using the DSF2 Walk Sequencer transform.

2.16.6.4.5 Becoming a DSF2 licensee

Before you can perform DSF2 processing in the software, you must complete the USPS DSF2 certification procedures and become licensed by the USPS.

Part of certification is processing test jobs in Data Services to prove that the software complies with the license agreement. When you are ready to take these tests, contact SAP BusinessObjects Business User Support to obtain access to the DSF2 features in Data Services.

Related Information

Reference Guide: DSF2 license process [page 1392]

2.16.6.4.6 DSF2 directories

DSF2 processing requires the following data:

Data	Notes
DPV directories	The software uses DPV directories to verify addresses and identify inaccurate addresses. SAP supplies the DPV directories with the U.S. National Directory delivery. Note DPV directories are included with the DSF2 tables. Do not use the DPV directories included with the DSF2 tables. Use the DPV directories from SAP with the U.S. National Directory delivery.
eLOT directories	The software uses eLOT directories to assign walk sequence numbers. SAP supplies the eLOT directories with the U.S. National Directory delivery. Note eLOT directories are included with the DSF2 tables. Do not use the eLOT directories included with the DSF2 tables. Use the eLOT directories from SAP with the U.S. National Directory delivery.
DSF2 tables	The software uses DSF2 tables to assign address attributes. Note DSF2 tables are supplied by the USPS and not SAP. In addition, the DSF2 tables include DPV and eLOT directories. Do not use the DPV and eLOT directories included with the DSF2 tables. Use the DPV and eLOT directories from SAP with the U.S. National Directory delivery.
Delivery statistics file	The software uses the delivery statistics file to provide counts of business and residential addresses per ZIP Code (Postcode1) per Carrier Route (Sort-

Data	Notes
	code). SAP supplies the delivery statistics file with the U.S. National Directory delivery.

You must specify the location of these directory files in the USA Regulatory Address Cleanse transform, except for the delivery statistics file. Set the location of the delivery statistics file (dsf.dir) in the DSF2 Walk Sequencer transform. Also, to meet DSF2 requirements, you must install updated directories monthly.

2.16.6.4.7 DSF2 augment processing

Set up DSF2 augment processing in the USA Regulatory Address Cleanse transform.

DSF2 processing requires DPV information, therefore, enable DPV in your job setup.

If you plan to use the output information from the DSF2 augment processing for walk sequence processing, you must also enable eLOT.

> ℹ Note
>
> DSF2 augment is available only in batch mode. You cannot add augment information to your data in real time.

2.16.6.4.7.1 DSF2 Augment directory expiration

The DSF2 directories are distributed monthly. You must use the DSF2 directories with U.S. National directories that are labeled for the same month. For example, the May 2011 DSF2 directories can be used with only the May 2011 National directories.

The DSF2 Augment data expires in 60 days instead of the 105 day expiration for the U.S. National directories. Because directories must all have the same base date (MM/YYYY), DSF2 users who have Yes set for the Enable DSF2 Augment option must update all of the U.S. National directories and other directories they use (for example, LACSLink or DPV) at the same time as the DSF2 Augment directories. The software will remind users to update the directories with a warning message that appears 15 days before the directory expires.

> ➔ Remember
>
> As with all directories, the software will not process your records with expired DSF2 directories.

2.16.6.4.7.2 Identify the DSF2 licensee

When you perform DSF2 processing, you must provide the following information: The DSF2-licensed company and the client for whom the company is processing this job.

You must complete the following options in the USPS License Information group for DSF2 processing:

- DSF2 Licensee ID
- Licensee Name
- List Owner NAICS Code
- List ID
- Customer Company Name
- Customer Company Address
- Customer Company Locality
- Customer Company Region
- Customer Company Postcode1
- Customer Company Postcode2
- List Received Date
- List Return Date

> ℹ Note
>
> If you are performing DSF2 and NCOALink processing in the same instance of the USA Regulatory Address Cleanse transform, then the information that you enter in the USPS License Information group must apply to both DSF2 and NCOALink processing. If, for example, the *List ID* is different for DSF2 and NCOALink, you will need to include two USA Regulatory Address Cleanse transforms: One for NCOALink and another for DSF2.

2.16.6.4.7.3 To enable DSF2 Augment

Before you can process with DSF2, you must first become a certified licensee.

In addition to the required customer company information that you enter into the USPS License Information group, set the following options to perform DSF2 Augment processing:

1. In the USA Regulatory Address Cleanse transform, open the *Options* tab.
2. Expand the Report and Analysis group and set the *Generate Report Data* option to Yes.
3. Expand the Reference Files group and enter the path for the options *DSF2 Augment Path*, *DPV Path*, and *eLOT Directory*, or use the $$RefFilesAddressCleanse substitution variable if you have it set up.
4. Also in the Reference Files group, enter a path for the *USPS Log Path* option, or use the $ $CertificationLogPath substitution variable if you have it set up.
5. Optional. Expand the Transform Performance group and set the *Cache DPV Directories* and *Cache DSF2 Augment Directories* to Yes.
6. Expand the Assignment Options group and set the *Enable DSF2 Augment*, *Enable DPV*, and *Enable eLOT* to Yes.
7. Include the DSF2 address attributes output fields in your output file setup.

2.16.6.4.7.4 DSF2 output fields

When you perform DSF2 Augment processing in the software, address attributes are available in the following output fields for every address that was assigned. Be sure to include the fields containing information you'll need in your output file setup:

- DSF2_Business_Indicator
- DSF2_Delivery_Type
- DSF2_Drop_Count
- DSF2_Drop_Indicator
- DSF2_Educational_Ind
- DSF2_LACS_Conversion_Ind
- DSF2_Record_Type
- DSF2_Seasonal_Indicator
- DSF2_Throwback_Indicator

> ℹ Note
>
> A blank output in any of these fields means that the address was not looked up in the DSF2 directories.

Related Information

Reference Guide: Data Quality fields, USA Regulatory Address Cleanse fields [page 1349]

2.16.6.4.7.5 Improve processing speed

You can cache DSF2 data to improve DSF2 processing speed.

To cache DSF2 data, Set the *Cache DSF2 Augment Directories* option in the Transform Performance group to Yes. The software caches only the directories needed for adding address attributes.

2.16.6.4.8 DSF2 walk sequencing

When you perform DSF2 walk sequencing in the software, the software adds delivery sequence information to your data, which you can use with presorting software to qualify for walk-sequence discounts.

> ➜ Remember
>
> The software does not place your data in walk sequence order.

Include the DSF2 Walk Sequencer transform to enable walk sequencing.

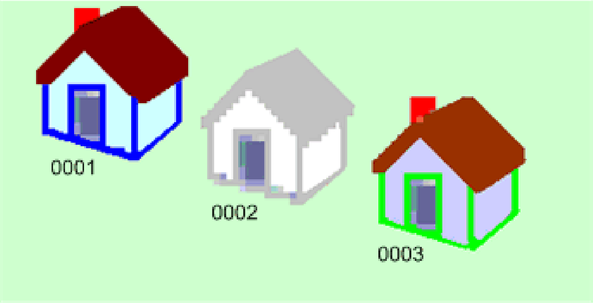
Related Information

Reference Guide: Transforms, Data Quality transforms, DSF2® Walk Sequencer [page 1172]

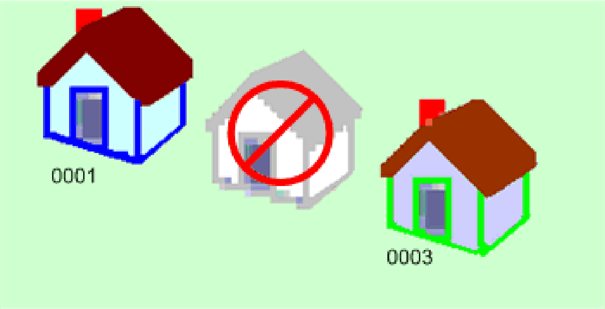
2.16.6.4.8.1 Pseudo sequencing

DSF2 walk sequencing is often called pseudo sequencing because it mimics USPS walk sequencing. Where USPS walk-sequence numbers cover every address, DSF2 processing provides pseudo sequence numbers for only the addresses in that particular file.

Walk sequencing includes every U.S. address, whether or not they're included in your list or mailing.



Pseudo sequencing includes only the addresses in your list. In the illustration below, assume the middle house is not included in your list.



The software uses DSF2 data to assign sequence numbers for all addresses that are DPV-confirmed delivery points (DPV_Status = Y). Other addresses present in your output file that are not valid DPV-confirmed delivery points (DPV_Status = S, N, or D) will receive "0000" as their sequence number. All other addresses will have a blank sequence number.

> Note
>
> When you walk-sequence your mail with the software, remember the following points:
>
> - (Batch only.) DSF2 walk sequencing is available only in batch mode. You cannot assign sequence numbers in real time.
> - Reprocess if you have made file changes. If your data changes in any way, you must re-assign sequence numbers. Sequence numbers are valid only for the data file as you process it at the time.

2.16.6.4.9 Break key creation

Break keys create manageable groups of data. They are created when there are two or more fields to compare.

The DSF2 Walk Sequencer transform automatically forms break groups before it adds walk sequence information to your data. The software creates break groups based on the Postcode1 and Sortcode_Route fields.

Set options for how you want the software to configure the fields in the Data Collection Config group. Keeping the default settings optimizes the data flow and allows the software to make the break key consistent throughout the data.

Option	Default value
Replace NULL with space	Yes

Option	Default value
Right pad with spaces	Yes

2.16.6.4.10 Enable DSF2 walk sequencing

To enable DSF2 walk sequence, include the DSF2 Walk Sequencer transform in your data flow.

2.16.6.4.10.1 Required information

When you set up for DSF2 walk sequence processing, the following options in the USPS License Information group are required:

- Licensee Name
- DSF2 Licensee ID
- List ID

2.16.6.4.10.2 To enable DSF2 walk sequencing

The input file for the DSF2 Walk Sequencer transform must have been pre-processed with CASS-certified software (such as the USA Regulatory Address Cleanse transform). To obtain an additional postage discount, include the DSF2_Business_Indicator output field information from CASS-certified software.

In addition to the required USPS License Information fields, make the following settings in the DSF2 Walk Sequencer transform:

1. Optional. Select Yes or No in the Common group, *Run as Separate Process* option. Select No if you are gathering DSF2 statistics. Select Yes to save processing time (if you don't need DSF2 statistics).
2. Enter the file path and file name (dsf.dir) to the Delivery Statistics directory in the *DelStats Directory* option in the Reference Files group. You may use the $$RefFilesAddressCleanse substitution parameter if you have it set up.
3. Enter the processing site location in the *Site Location* option of the Processing Options group. This is applicable only if you have more than one site location for DSF2 processing.
4. Make the following settings in the Data Collection Configuration group:
   - Select Yes or No in the *Replace Null With Space* option as desired.
   - Select Yes or No for the *Right Pad With Spaces* option as desired.
   - Select Yes or No for the *Pre Sorted Data* option (optional). We recommend that you keep the default setting of No so that Data Services sorts your data based on the break key fields (instead of using another software program).

2.16.6.4.11 DSF2 walk sequence input fields

Here is a list of the DSF2 walk sequence input fields.

> Note
>
> These fields must have been output from CASS-certified software processing before they can be used as input for the DSF2 Walk Sequencer transform:

- Postcode1
- Postcode2
- Sortcode_Route
- LOT
- LOT_Order
- Delivery_Point
- DPV_Status
- DSF2_Business_Indicator (optional)

The software uses the information in these fields to determine the way the records should be ordered (walk sequenced) if they were used in a mailing list. The software doesn't physically change the order of your database. The software assigns walk-sequence numbers to each record based on the information it gathers from these input fields.

> Note
>
> All fields are required except for the DSF2_Business_Indicator field.

The optional DSF2_Business_Indicator field helps the software determine if the record qualifies for saturation discounts. Saturation discounts are determined by the percentage of residential addresses in each carrier route. See the *USPS Domestic Mail Manual* for details about all aspects of business mailing and sorting discounts.

Related Information

Reference Guide: Transforms, DSF2® Walk Sequencer, Input fields [page 1176]

2.16.6.4.12 DSF2 walk-sequence output fields

The software outputs walk-sequence number information to the following fields:

- Active_Del_Discount
- Residential_Sat_Discount
- Sortcode_Route_Discount
- Walk_Sequence_Discount
- Walk_Sequence_Number

Related Information

Reference Guide: Data Quality fields, DSF2 Walk Sequencer, DSF2 Walk Sequencer output fields [page 1177]

2.16.6.4.13 DSF2 reporting

There are reports and log files that the software generates for DSF2 augment and walk sequencing.

Find complete information about these reports and log files in the *Management Console Guide*.

Delivery Sequence Invoice Report

The USPS requires that you submit the Delivery Sequence Invoice report if you claim DSF2 walk-sequence discounts for this job.

US Addressing Report

- The US Addressing Report is generated by the USA Regulatory Address Cleanse transform.
- The Second Generation Delivery Sequence File Summary and Address Delivery Types sections of the US Addressing Report shows counts and percentages of addresses in your file that match the various DSF2 categories (if NCOALink is enabled). The information is listed for pre and post NCOALink processing.

DSF2 Augment Statistics Log File

The USPS requires that DSF2 licensees save information about their processing in the DSF2 log file. The USPS dictates the contents of the DSF2 log file and requires that you submit it to them monthly.

Log files are available to users with administrator or operator permissions.

Related Information

Management Console Guide: Administrator, Administrator management, Exporting DSF2 certification log [page 1877]
Management Console Guide: Data Quality reports, Delivery Sequence Invoice Report [page 2001]
Management Console Guide: Data Quality reports, US Addressing Report [page 2003]

2.16.6.4.13.1 DSF2 Augment Statistics Log File

The DSF2 Augment Statistics Log File is stored in the repository. The software generates the log file to the repository where you can export them by using the Data Services Management Console (for Administrators or Operators only).

The naming format for the log file is as follows:

```
[DSF2_licensee_ID][mm][yy].dat
```

The USPS dictates the contents of the DSF2 log file and requires that you submit it to them monthly. For details, read the *DSF2 Licensee Performance Requirements* document, which is available on the USPS RIBBS website (*http://ribbs.usps.gov/dsf2/documents/tech_guides*).

You must submit the DSF2 log file to the USPS by the third business day of each month by e-mail.

Log file retention and automatic deletion

You must submit the Augment Statistics Log File to the USPS every month. The software deletes log files on a periodic basis (default is 30 days), which can be controlled through Data Services Application Settings in the Central Management Console. To avoid losing monthly log information, set the *History Retention Period* to more than 31 days (we recommend a setting of 50 days).

In addition to sending monthly log files to the USPS, you are required to have the data available for the USPS to examine for several years after the job is processed. (Make sure you are aware of current USPS rules for data retention by checking your USPS licensing agreement.) To ensure that you retain all required reports and logs before the data is deleted from the repository, we recommend that you export the required reports and logs from the repository to a local folder on a monthly basis. This also prevents the repository contents from becoming so large that the export process “times out” due to the volume of statistics retained.

Related Information

Administrators Guide: USPS log files and reports [page 59]
Administrators Guide: Setting the history retention period [page 58]

2.16.6.5 NCOALink® overview

The USPS Move Update standard helps users and the USPS to reduce the number of records that are returned because the address is out of date. NCOALink is a part of this effort. Move Updating is the process of checking addresses against the National Change of Address (NCOA) database to make sure your data is updated with current addresses.

When you process your data using NCOALink, you update your records for individuals or businesses that have moved and have filed a Change of Address (COA) form with the USPS. Other programs that are a part of Move

Update, and that are supported in the USA Regulatory Address Cleanse transform, include ANKLink®, and SuiteLink®.

The USPS requires that your lists comply with Move Update standards in order for it to qualify for the discounted postal rates available for First-Class presorted mailings. You can meet this requirement through the NCOALink process.

> Note
>
> Mover ID is the name under which SAP Data Services is certified for NCOALink.

Related Information

ANKLink overview [page 655]
SuiteLink™ [page 637]

2.16.6.5.1 The importance of move updating

The USPS requires move updating on all First Class presorted mailings. To help mailers meet this requirement, the USPS offers certain options, including NCOALink.

To keep accurate address information for your contacts, you must use a USPS method for receiving your contacts' new addresses. Not only is move updating good business, it is required for all First-Class mailers who claim presorted or automation rates. As the USPS expands move-updating requirements and more strictly enforces the existing regulations, move updating will become increasingly important.

Related Information

ANKLInk [page 655]

2.16.6.5.2 Benefits of NCOALink

By using NCOALink in the USA Regulatory Address Cleanse transform, you are updating the addresses in your lists with the latest move data. With NCOALink, you can:

- Improve mail deliverability.
- Reduce the cost and time needed to forward mail.
- Meet the USPS move-updating requirement for presorted First Class mail.
- Prepare for the possible expansion of move-update requirements.

2.16.6.5.3 How NCOALink works

When processing addresses with NCOALink enabled, the software follows these steps:

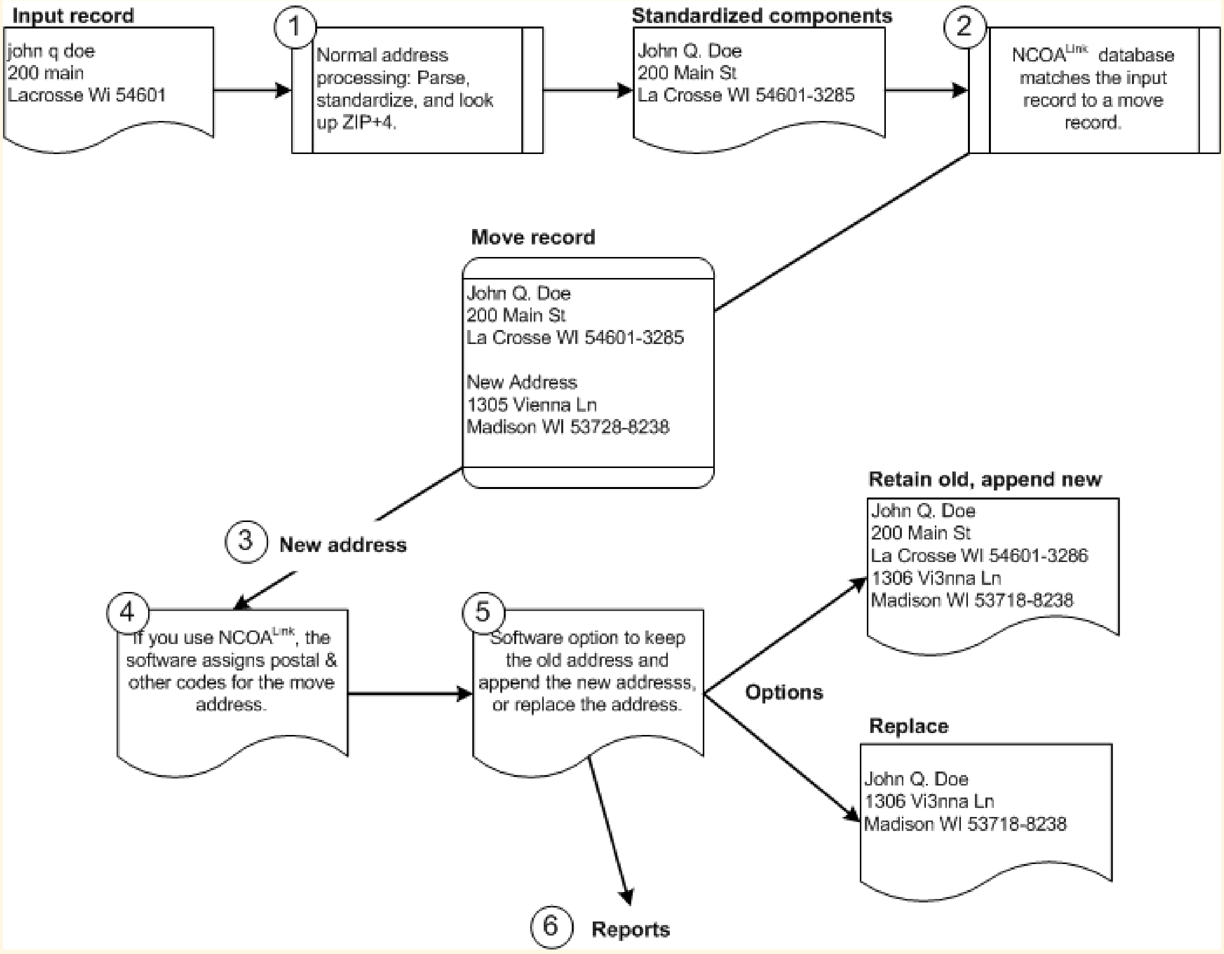
1. The USA Regulatory Address Cleanse transform standardizes the input addresses. NCOALink requires parsed, standardized address data as input.
2. The software searches the NCOALink database for records that match your parsed, standardized records.
3. If a match is found, the software receives the move information, including the new address, if one is available.
4. The software looks up move records that come back from the NCOALink database to assign postal and other codes.
5. Depending on your field class selection, the output file contains:
    - The original input address. The complete and correct value found in the directories, standardized according to any settings that you defined in the Standardization Options group in the Options tab. (CORRECT)
    - The address components that have been updated with move-updated address data. (MOVE-UPDATED)

      Note

      The transform looks for the move-updated address information in the U.S. National Directories. When the move-updated address is not found in the U.S. National Directories, the software populates the Move Updated fields with information found in the Move Update Directories only. The Move Updated fields that are populated as a result of standardizing against the U.S. National Directories will not be updated.
    - The move-updated address data if it exists and if it matches in the U.S. National directories. Or the field contains the original address data if a move does not exist or if the move does not match in the U.S. National Directories. (BEST)

    Based on the *Apply Move to Standardized Fields* option in the NCOALink group, standardized components can contain either original or move-updated addresses.
6. The software produces the reports and log files required for USPS compliance.

Example



1. NCOALink requires parsed, standardized address data as input. Therefore, before NCOALink processing, the software performs its normal processing on the address data.
2. The software searches the NCOALink database for a record that matches your parsed, standardized record.
3. The software receives the move information, including the new address if one is available.
4. The software looks up the move record that comes back from the NCOALink database, to assign postal and other codes.
5. At your option, the software can either retain the old address and append the new, or replace the old address with the new.
6. The software produces the reports and log files that you will need for USPS compliance.

2.16.6.5.4 NCOALink provider levels

NCOALink users fall in one of three categories of providers. Specify the service provider in the USPS License Information group of options under *Provider Level*.

Note

Only provider levels supported in your registered keycodes display in the selection list.

Provider level	Description
Full Service Provider (FSP)	Provides NCOALink processing to third parties.
Limited Service Provider (LSP)	Provides NCOALink processing to third parties and internally.
End User Mailer (EUM)	Provides NCOALInk processing to in-house lists only.

2.16.6.5.5 NCOALink brokers and list administrators

An NCOALink user may have a broker or list administrator who owns the lists they are processing. When there is a broker or list administrator involved, add contact information in the NCOALink group under *Contact Detail list* › *Contact Details*.

Broker

A broker directs business to an NCOALink service provider.

List administrator

A list administrator maintains and stores lists. List administrators are different than brokers in two ways:

- List administrators don't send move-updated files back to the list owner.
- List administrators may have an NCOALink license.

If a list administrator, a broker, or both are involved in your job, you must complete Contact Detail List for each of them separately. You can duplicate a group of options by right-clicking the group name and choosing *Duplicate Option*.

2.16.6.5.6 Address not known (ANKLink)

Undeliverable-as-addressed (UAA) mail costs the mailing industry and the USPS a lot of money each year. The software provides NCOALink as an additional solution to UAA mail. With NCOALink, you also can have access to the USPS's ANKLink data.

2.16.6.5.6.1 About ANKLink

NCOALink limited service providers and end users receive change of address data for the preceding 18 months. The ANKLink option enhances that information by providing additional data about moves that occurred in the previous months 19 through 48.

> ➜ Tip
>
> If you are an NCOALink full service provider you already have access to the full 48 months of move data (including the new addresses).

> i Note
>
> The additional 30 months of data that comes with ANKLink indicates only that a move occurred and the date of the move; the new address is not provided.

The ANKLink data helps you make informed choices regarding a contact. If the data indicates that the contact has moved, you can choose to suppress that contact from the list or try to acquire the new address from an NCOALINK full service provider.

If you choose to purchase ANKLink to extend NCOALINK information, then the DVD you receive from the USPS will contain both the NCOALink 18-month full change of address information and the additional 30 month ANKLink information which indicates that a move has occurred.

If an ANKLink match exists, it is noted in the ANKLINK_RETURN_CODE output field and in the NCOALink Processing Summary report.

2.16.6.5.6.2 ANKLink data

ANKLink is a subset of NCOALink. You can request ANKLink data from the USPS National Customer Support Center (NCSC) by calling 1-800-589-5766 or by e-mail at ncoalink@usps.gov. ANKLink data is not available from SAP.

The software detects if you're using ANKLink data. Therefore, you do not have to specify whether you're using ANKLink in your job setup.

2.16.6.5.6.3 ANKLink support for NCOALink provider levels

The software supports three NCOALink provider levels defined by the USPS. Software options vary by provider level and are activated based on the software package that you purchased. The following table shows the provider levels and support:

Provider level	Provide service to third parties	COA data (months)	Data received from USPS	Support for ANKLink
Full Service Provider (FSP)	Yes Third party services must be at least 51% of all processing.	48	Weekly	No (no benefit)
Limited Service Provider (LSP)	Yes LSPs can both provide services to third parties and use the product internally.	18	Weekly	Yes
End User Mailer (EUM)	No	18	Monthly	Yes

➡ Tip

If you are an NCOALink EUM, you may request a stop processing alternative agreement from the USPS. After you are approved by the USPS you may purchase the software's stop processing alternative functionality which allows DPV and LACSLink processing to continue after a false positive address record is detected.

Related Information

Stop Processing Alternative Agreement [page 620]

2.16.6.5.7 Software performance

In our tests, the software ran slower with NCOALink enabled than with it disabled. Your processing speed depends on the computer running the software and the percentage of input records affected by a move (more moves equals slower performance).

Related Information

Improve NCOALink processing performance [page 660]

2.16.6.5.8 Getting started with NCOALink

Before you begin NCOALink processing you need to perform the following tasks:

- Complete the USPS certification process to become an NCOALink service provider or end user. For information about certification, see the NCOALink Certification section following the link below.
- Understand the available output strategies and performance optimization options.
- Configure your job.

2.16.6.5.9 What to expect from the USPS and SAP

NCOALink, and the license requirements that go with it, has created a new dimension in the relationship among mailers (you), the USPS, and vendors. It's important to be clear about what to expect from everyone.

2.16.6.5.9.1 Move updating is a business decision for you to make

NCOALink offers an option to replace a person's old address with their new address. You as a service provider must decide whether you accept move updates related to family moves, or only individual moves. The USPS recommends that you make these choices only after careful thought about your customer relationships. Consider the following examples:

- If you are mailing checks, account statements, or other correspondence for which you have a fiduciary responsibility, then move updating is a serious undertaking. The USPS recommends that you verify each move by sending a double postcard, or other easy-reply piece, before changing a financial record to the new address.
- If your business relationship is with one spouse and not the other, then move updating must be handled carefully with respect to divorce or separation. Again, it may make sense for you to take the extra time and expense of confirming each move before permanently updating the record.

2.16.6.5.9.2 NCOALink security requirements

Because of the sensitivity and confidentiality of change-of-address data, the USPS imposes strict security procedures on software vendors who use and provide NCOALink processing.

One of the software vendor's responsibilities is to check that each list input to the USA Regulatory Address Cleanse transform contains at least 100 unique records. Therefore the USA Regulatory Address Cleanse transform checks your input file for at least 100 unique records. These checks make verification take longer, but they are required by the USPS and they must be performed.

If the software finds that your data does not have 100 unique records, it issues an error and discontinues processing.

The process of checking for 100 unique records is a pre-processing step. So if the software does not find 100 unique records, there will be no statistics output or any processing performed on the input file.

Related Information

Desinger Guide: Getting started with NCOALink [page 656]

2.16.6.5.9.2.1 How the software checks for 100 unique records

When you have NCOALink enabled in your job, the software checks for 100 unique records before any processing is performed on the data. The software checks the entire database for 100 unique records. If it finds 100 unique records, the job is processed as usual. However, if the software does not find 100 unique records, it issues an error stating that your input data does not have 100 unique records, or that there is not enough records to determine uniqueness.

For the 100 unique record search, a record consists of all mapped input fields concatenated in the same order as they are mapped in the transform. Each record must be identical to another record for it to be considered alike (not unique).

Example

Comparing records

The example below illustrates how the software concatenates the fields in each record, and determines non-unique records. The first and last row in this example are not unique.

332	FRONT	STREET	NORTH	LACROSSE	WI	54601
332	FRONT	STREET	SOUTH	LACROSSE	WI	54601
331	FRONT	STREET	SOUTH	LACROSSE	WI	54601
332	FRONT	STREET	NORTH	LACROSSE	WI	54601

2.16.6.5.9.2.2 Finding unique records in multiple threads

Sometimes input list have 100 unique records but the user still receives an error message stating that the list does not have 100 unique records. This can happen when there is a low volume of data in lists. To work around this problem, users can adjust the Degree of Parallelism (DOP) setting in their job.

Low volume of data and DOP > 1

When an NCOALink job is set up with the DOP greater than 1, each thread checks for unique records within the first collection it processes and shares knowledge of the unique records it found with all other threads. The first thread to finish processing it's collection counts the unique records found by all threads up to that point in time and makes a decision regarding whether or not the 100 record minimum check has been satisfied. That thread may not necessarily be thread 1. For example, say your list has 3,050 records and you have the DOP set for 4. If the number of records per collection is 1000, each thread will have a collection of 1000 records except for the last thread which will only have 50 records. The thread processing 50 records is likely to finish its collection sooner and it may make the pass/fail decision before 100 unique records have been encountered. You may be able to successfully run this job if you lower the DOP. In this example, you could lower it to 3.

2.16.6.5.10 About the NCOALink daily delete file

If you are a service provider, then every day before you perform NCOALink processing, you must download the daily delete file and install it in the same folder that your NCOALink directories are located.

The daily delete file contains records that are pending deletion from the NCOALink data. For example, if Jane Doe filed a change of address with the USPS and then didn’t move, Jane’s record would be in the daily delete file. Because the change of address is stored in the NCOALink directories, and they are updated only weekly or monthly, the daily delete file is needed in the interim, until the NCOALink directories are updated again.

Download the daily delete file from the USPS's electronic product fulfillment website at *https://epf.usps.gov/* . Before you can access the file, you must complete and submit the Electronic Product Fulfillment Web Access Request Form (Form 5116) to the USPS. You can obtain this form from the same website.

> Note
>
> If you are an end user, you only need the daily delete file for processing Stage I or II files. It is not required for normal NCOALink processing.

Here are some important points to know about the daily delete file:

- The software will fail verification if NCOALink is enabled, a stage test is being performed, and the daily delete file isn't installed.
- USA Regulatory Address Cleanse transform supports only the ASCII version of the daily delete file.
- Do not rename the daily delete file. It must be named `dailydel.dat`.
- The software will issue a verification warning if the daily delete file is more than three days old.

2.16.6.5.10.1 To install the NCOALink daily delete file

To download and install the NCOALink daily delete file, follow these steps:

1. Go to the USPS Electronic Product Fulfillment site at *https://epf.usps.gov/* .
2. Before you download the daily delete file for the first time, you must complete and fax the PS Form 5116 (Electronic Product Fulfillment Web Access Request Form) to the USPS Licensing Department.

   When completing the form, make sure that you select the NCOALink or NCOALink with ANKLink option, as appropriate. This allows you to access the daily delete file.
3. Log into the USPS Electronic Product Fulfillment site and download the NCOALink Daily Delete [TEXT] file to a location where the `.tar` file can be extracted. If your computer browser has pop-up blockers enabled, you may need to override them.

4. Extract the `dailyDeletes_txt.tar` file.
5. Copy the `dailydel.dat` file to the same location where your NCOALink directories are stored.
6. Repeat steps 3–5 every day before you perform NCOALink processing.

2.16.6.5.11 Output file strategies

You can configure your output file to meet your needs. Depending on the Field Class Selection that you choose, components in your output file contain Correct, Move-updated, or Best information:

- CORRECT: Outputs the original input address. The complete and correct value found in the directories, standardized according to any settings that you defined in the Standardization Options group in the Options tab. (CORRECT)
- MOVE-UPDATED: Outputs the address components that have been updated with move-updated address data.

  > Note
  >
  > The transform looks for the move-updated address information in the U.S. National Directories. When the move-updated address is not found in the U.S. National Directories, the software populates the Move Updated fields with information found in the Move Update Directories only. The Move Updated fields that are populated as a result of standardizing against the U.S. National Directories will not be updated.

- BEST: Outputs the move-updated address data if it exists and if it matches in the U.S. National directories. Or the field contains the original address data if a move does not exist or if the move does not match in the U.S. National Directories.

Based on the *Apply Move to Standardized Fields* option setting in the NCOA option group, standardized components can contain original or move-updated addresses.

By default the output option *Apply Move to Standardized Fields* is set to Yes and the software updates standardized fields to contain details about the updated address available through NCOALink.

If you want to retain the old addresses in the standardized components and append the new ones to the output file, you must change the *Apply Move to Standardized Fields* option to No. Then you can use output fields such as NCOALINK_RETURN_CODE to determine whether a move occurred. One way to set up your output file is to replicate the input file format, then append extra fields for move data. In the output records not affected by a move, most of the appended fields will be blank. Alternatively, you can create a second output file specifically for move records. Two approaches are possible:

- Output each record once, placing move records in the second output file and all other records in the main output file.
- Output move records twice; once to the main output file, and a second time to the second output file.

Both of these approaches require that you use an output filter to determine whether a record is a move.

2.16.6.5.12 Improving NCOALink processing performance

Many factors affect performance when processing NCOALink data. Generally the most critical factor is the volume of disk access that occurs. Often the most effective way to reduce disk access is to have sufficient memory available to cache data. Other critical factors that affect performance include hard drive speed, seek time, and the sustained transfer rate. When the time spent on disk access is minimized, the performance of the CPU becomes significant.

Related Information

Finding unique records in multiple threads [page 658]

2.16.6.5.12.1 Operating systems and processors

The computation involved in most of the software and NCOALink processing is very well-suited to the microprocessors found in most computers, such as those made by Intel and AMD. RISC style processors like those found in most UNIX systems are generally substantially slower for this type of computation. In fact a common PC can often run a single job through the software and NCOALink about twice as fast as a common UNIX system. If you're looking for a cost-effective way of processing single jobs, a Windows server or a fast workstation can produce excellent results. Most UNIX systems have multiple processors and are at their best processing several jobs at once.

You should be able to increase the degree of parallelism (DOP) in the data flow properties to maximize the processor or core usage on your system. Increasing the DOP depends on the complexity of the data flow.

2.16.6.5.12.2 Memory

NCOALink processing uses many gigabytes of data. The exact amount depends on your service provider level, the data format, and the specific release of the data from the USPS.

In general, if performance is critical, and especially if you are an NCOALink full service provider and you frequently run very large jobs with millions of records, you should obtain as much memory as possible. You may want to go as far as caching the entire NCOALink data set. You should be able to cache the entire NCOALink data set using 20 GB of RAM, with enough memory left for the operating system.

2.16.6.5.12.3 Data storage

If at all possible, the hard drive you use for NCOALink data should be fully dedicated to that process, at least while your job is running. Other processes competing for the use of the same physical disk drive can greatly reduce your NCOALink performance.

To achieve even higher transfer rates you may want to explore the possibility of using a RAID system (redundant array of independent discs).

When the software accesses NCOALink data directly instead of from a cache, the most significant hard drive feature is the average seek time.

2.16.6.5.12.4 Data format

The software supports both hash and flat file versions of NCOALink data. If you have ample memory to cache the entire hash file data set, that format may provide the best performance. The flat file data is significantly smaller, which means a larger share can be cached in a given amount of RAM. However, accessing the flat file data involves binary searches, which are slightly more time consuming than the direct access used with the hash file format.

2.16.6.5.12.5 Memory usage

The optimal amount of memory depends on a great many factors. The “Auto” option usually does a good job of deciding how much memory to use, but in some cases manually adjusting the amount can be worthwhile.

2.16.6.5.12.6 Performance tips

Many factors can increase or decrease NCOALink processing speed. Some are within your control and others may be inherent to your business. Consider the following factors:

- Cache size—Using too little memory for NCOALink caching can cause unnecessary random file access and time-consuming hard drive seeks. Using far too much memory can cause large files to be read from the disk into the cache even when only a tiny fraction of the data will ever be used. The amount of cache that works best in your environment may require some testing to see what works best for your configuration and typical job size.
- Directory location—It’s best to have NCOALink directories on a local solid state drive or a virtual RAM drive. Using a local solid state drive or virtual RAM drive eliminates all I/O for NCOALink while processing your job. If you have the directories on a hard drive, it’s best to use a defragmented local hard drive. The hard drive should not be accessed for anything other than the NCOALink data while you are running your job.
- Match rate—The more records you process that have forwardable moves, the slower your processing will be. Retrieving and decoding the new addresses takes time, so updating a mailing list regularly will improve the processing speed on that list.
- Input format—Ideally you should provide the USA Regulatory Address Cleanse transform with discrete fields for the addressee’s first, middle, and last name, as well as for the pre-name and post-name. If your input has only a name line, the transform will have to take time to parse it before checking NCOALink data.
- File size—Larger files process relatively faster than smaller files. There is overhead when processing any job, but if a job includes millions of records, a few seconds of overhead becomes insignificant.

2.16.6.5.13 To enable NCOALink processing

You must have access to the following files:

- NCOALink directories
- Current version of the USPS daily delete file
- DPV data files

- LACSLink data files

If you use a copy of the sample transform configuration, USARegulatoryNCOALink_AddressCleanse, NCOALink, DPV, and LACSLink are already enabled.

1. Open the *USA Regulatory Address Cleanse* transform and open the *Options* tab.
2. Set values for the options as appropriate for your situation.

   For more information about the USA Regulatory Address Cleanse transform fields, see the *Reference Guide*. The table below shows fields that are required only for specific provider levels.

Option group	Option name or subgroup	End user without stop processing alternative agreement	End user with stop processing alternative agreement	Full or limited service provider
	Licensee Name	yes	yes	yes
	List Owner NAICS Code	yes	yes	yes
USPS License Information	*List ID*	no	no	yes
	Customer Company Name	no	yes	yes
	Customer Company Address	no	yes	yes
	Customer Company Locality	no	yes	yes
	Customer Company Region	no	yes	yes
	Customer Company Postcode1	no	yes	yes
	Customer Company Postcode2	no	yes	yes
	Customer Company Phone	no	no	no
	List Processing Frequency	yes	yes	yes
	List Received Date	no	no	yes
	List Return Date	no	no	yes
	Provider Level	yes	yes	yes
NCOALink	*PAF Details subgroup*	no	no	All options are required, except Customer Parent Company Name and Customer Alternate Company Name.
	Service Provider Options subgroup	no	no	All options are required, except Buyer Company Name and Postcode for Mail Entry.

➜ Tip

If you are a service provider and you need to provide contact details for multiple brokers, expand the NCOALink group, right-click *Contact Details* and click *Duplicate Option*. An additional group of contact detail fields will be added below the original group.

Related Information

Reference Guide: USA Regulatory Address Cleanse transform [page 1317]
Reference Guide: About NCOA directories [page 1386]
About the NCOALink daily delete file [page 659]
Output file strategies [page 660]
Stop Processing Alternative [page 620]

2.16.6.5.14 NCOALink log files

The software automatically generates the USPS-required log files and names them according to USPS requirements. The software generates these log files to the repository where you can export them by using the Management Console.

The software creates one log file per license ID. At the beginning of each month, the software starts new log files. Each log file is then appended with information about every NCOALink job processed that month for that specific license ID.

The software produces the following move-related log files:

- CSL (Customer Service log)
- PAF (Processing Acknowlagement Form) customer Information log
- BALA (Broker/Agent/List Administrator) log

The PAF Customer Information Log File and the BALA Log File are not required for end users.The following table shows the log files required for Limited or Full Service Providers:

Log file	Description
CSL	This log file contains one record per list that you process. Each record details the results of change-of-address processing.
PAF customer information log	This log file contains the information that you provided for the PAF. The log file lists each unique PAF entry. If a list is processed with the same PAF information, the information appears just once in the log file. When contact information for the list administrator has changed, then information for both the list administrator and the corresponding broker are written to the PAF log file.

Log file	Description
BALA	This log file contains all of the contact information that you entered for the broker or list administrator. The log file lists information for each broker or list administrator just once. The USPS requires the Broker/Agent/List Administrator log file from service providers, even in jobs that do not involve a broker or list administrator. The software produces this log file for every job if you're a certified service provider.

Log file retention and automatic deletion

You must submit the NCOALink log files to the USPS every month. The software deletes log files on a periodic basis (default is 30 days), which can be controlled through Data Services Application Settings in the Central Management Console. To avoid losing monthly log information, set the *History Retention Period* to more than 31 days (we recommend a setting of 50 days).

In addition to sending monthly log files to the USPS, you are required to have the data available for the USPS to examine for several years after the job is processed. (Make sure you are aware of current USPS rules for data retention by checking your USPS licensing agreement.) To ensure that you retain all required reports and logs before the data is deleted from the repository, we recommend that you export the required reports and logs from the repository to a local folder on a monthly basis. This also prevents the repository contents from becoming so large that the export process “times out” due to the volume of statistics retained.

Related Information

Management Console Guide: NCOALink Processing Summary Report [page 2000]
Management Console Guide: Exporting NCOALink certification logs [page 1878]
Administrators Guide: USPS log files and reports [page 59]
Administrators Guide: Setting the history retention period [page 58]

2.16.6.5.14.1 Log file names

The software follows the USPS file-naming scheme for the following log files:

- Customer Service log
- PAF Customer Information log
- Broker/Agent/List Administrators log

The table below describes the naming scheme for NCOALink log files. For example, `P1234C10.DAT` is a PAF Log file generated in December 2010 for a licensee with the ID 1234.

Character 1		Characters 2 -5	Character 6		Characters 7-8	Extension
Log type		Platform ID	Month		Year	
B	Broker log	Exactly four characters long	1	January	Two characters , for example 10 for 2010	.DAT
C	Customer service log		2	February		
P	PAF log		3	March		
			4	April		
			5	May		
			6	June		
			7	July		
			8	August		
			9	September		
			A	October		
			B	November		
			C	December		

For example, `P1234C10.DAT` is a PAF Log file generated in December 2010 for a licensee with the ID 1234.

2.16.6.6 USPS eLOT®

eLOT is available for U.S. records in the USA Regulatory Address Cleanse transform only.

eLOT takes line of travel one step further. The original LOT narrowed the mail carrier's delivery route to the block face level (Postcode2 level) by discerning whether an address resided on the odd or even side of a street or thoroughfare.

eLOT narrows the mail carrier's delivery route walk sequence to the house (delivery point) level. This allows you to sort your mailings to a more precise level.

Related Information

To enable eLOT [page 667]
Set up the reference files [page 594]

2.16.6.6.1 To enable eLOT

1. Open the USA Regulatory Address Cleanse transform.
2. Open the *Options* tab, expand the Assignment Options group, and select Yes for the *Enable eLOT* option.
3. In the *Reference Files* group, set the path for your eLOT directory.

   You can use the subtitution variable $$RefFilesAddressCleanse for this option if you have it set up.

2.16.6.7 Early Warning System (EWS)

EWS helps reduce the amount of misdirected mail caused when valid delivery points are created between national directory updates. EWS is available for U.S. records in the USA Regulatory Address Cleanse transform only.

2.16.6.7.1 Overview of EWS

The EWS feature is the solution to the problem of misdirected mail caused by valid delivery points that appear between national directory updates. For example, suppose that 300 Main Street is a valid address and that 300 Main Avenue does not exist. A mail piece addressed to 300 Main Avenue is assigned to 300 Main Street on the assumption that the sender is mistaken about the correct suffix.

Now consider that construction is completed on a house at 300 Main Avenue. The new owner signs up for utilities and mail, but it may take a couple of months before the delivery point is listed in the national directory. All the mail intended for the new house at 300 Main Avenue will be mis-directed to 300 Main Street until the delivery point is added to the national directory.

The EWS feature solves this problem by using an additional directory which informs CASS users of the existence of 300 Main Avenue long before it appears in the national directory. When using EWS processing, the previously mis-directed address now defaults to a 5-digit assignment.

2.16.6.7.2 Start with a sample transform configuration

If you want to use the USA Regulatory Address Cleanse transform with the EWS option turned on, it is best to start with the sample transform configuration for EWS processing named USARegulatoryEWS_AddressCleanse.

2.16.6.7.3 EWS directory

The EWS directory contains four months of rolling data. Each week, the USPS adds new data and drops a week's worth of old data. The USPS then publishes the latest EWS data. Each Friday, SAP converts the data to our format (`EWyymmdd.zip`) and posts it on the SAP Buisiness User Support site at *https://service.sap.com/bosap-downloads-usps*.

2.16.6.7.4 To enable EWS

EWS is already enabled when you use the software's EWS sample transform, USARegulatoryEWS_AddressCleanse. These steps show how to manually set EWS.

1. Open the USA Regulatory Address Cleanse transform.
2. Open the *Options* tab and expand the Assignment Options group.
3. Select Enable for the *Enable EWS* option.
4. Expand the Reference Files group and enter a path for the *EWS Directory* option, or use the substitution variable $$RefFilesAddressCleanse if you have it set up.

Related Information

Early Warning System (EWS) [page 667]

2.16.6.8 USPS RDI®

The RDI (Residential Delivery Indicator) option is available in the USA Regulatory Address Cleanse transform. RDI determines whether a given address is for a residence or non residence.

Parcel shippers can find RDI information to be very valuable because some delivery services charge higher rates to deliver to residential addresses. The USPS, on the other hand, does not add surcharges for residential deliveries. When you can recognize an address as a residence, you have increased incentive to ship the parcel with the USPS instead of with a competitor that applies a residential surcharge.

According to the USPS, 91-percent of U.S. addresses are residential. The USPS is motivated to encourage the use of RDI by parcel mailers.

You can use RDI if you are processing your data for CASS certification or if you are processing in a non-certified mode. In addition, RDI does not require that you use DPV processing.

2.16.6.8.1 Start with a sample transform

If you want to use the RDI feature with the USA Regulatory Address Cleanse transform, it is best to start with the sample transform configuration, USARegulatoryRDI_AddressCleanse.

Sample transforms are located in the Transforms tab of the Object Library. This sample is located under USA_Regulatory_Address_Cleanse transforms.

2.16.6.8.2 How RDI works

After you install the RDI directories and enable RDI processing, the software determines whether the address represented by an 11-digit postcode (Postcode1, Postcode2, and the DPBC) is a residential address. (The software can sometimes do the same with a postcode2.)

The software indicates whether an address is for a residence in the output component, RDI_INDICATOR.

Using the RDI feature involves only a few steps:

1. Install the RDI directories.
2. Specify where the directories are located.
3. Enable RDI processing in the software.
4. Run the job.

Related Information

Enable RDI [page 670]

2.16.6.8.2.1 Compatibility

RDI has the following compatibility with other options in the software:

- RDI is allowed in both CASS and non-CASS processing modes.
- RDI is allowed with or without DPV processing.

2.16.6.8.3 RDI directory files

SAP ships the RDI directory files with the U.S. National Directory update.

RDI requires the following directories:

File	Description
`rts.hs11`	For 11-digit postcode lookups (Postcode2 plus DPBC). This file is used when an address contains an 11-digit postcode. Determination is based on the delivery point.
`rts.hs9`	For 9-digit postcode lookups (Postcode2). This file is based on a ZIP+4.This is possible only when the addresses for that ZIP+4 are for all residences or for no residences.

2.16.6.8.3.1 Specify RDI directory path

In the Reference Files group, specify the location of your RDI directories in the *RDI Path* option. If RDI processing is disabled, the software ignores the *RDI Path* setting.

2.16.6.8.4 To enable RDI

If you use a copy of the USARegulatoryRDI_AddressCleanse sample transform in your data flow, RDI is already enabled. However, if you are starting from a USA Regulatory Address Cleanse transform, make sure you enable RDI and set the location for the following RDI directories: `rts.hs11` and `rts.hs9`.

1. Open the *USA Regulatory Address Cleanse* transform.
2. In the *Options* tab expand the Reference Files group, and enter the location of the RDI directories in the *RDI Path* option, or use the substitution variable $$RefFilesAddressCleanse if you have it set up.
3. Expand the Assignment Options group, and select *Yes* for the *Enable RDI* option.

2.16.6.8.5 RDI output field

For RDI, the software uses a single output component that is always one character in length. The RDI component is populated only when the *Enable RDI* option in the Assignment Options group is set to Yes.

Job/Views field	Length	Description
RDI_INDICATOR	1	This field contains the RDI value that consists of one of the following values: Y = The address is for a residence. N = The address is not for a residence.

2.16.6.8.6 RDI in reports

A few of the software's reports have additional information because of the RDI feature.

2.16.6.8.6.1 CASS Statement, USPS Form 3553

The USPS Form 3553 contains an entry for the number of residences. (The CASS header record also contains this information.)

2.16.6.8.6.2 Statistics files

The statistics file contains RDI counts and percentages.

2.16.6.9 GeoCensus (USA Regulatory Address Cleanse)

The GeoCensus option of the USA Regulatory Address Cleanse transform offers geographic and census coding for enhanced sales and marketing analysis. It is available for U.S. records only.

> ℹ Note
>
> GeoCensus functionality in the USA Regulatory Address Cleanse transform will be deprecated in a future version. It is recommended that you upgrade any data flows that currently use the GeoCensus functionality to use the Geocoder transform. For instructions on upgrading from GeoCensus to the Geocoder transform, see the *Upgrade Guide*.

Related Information

How GeoCensus works [page 671]
GeoCensus data and directories [page 673]
To enable GeoCensus coding [page 674]
Geocoding [page 489]

2.16.6.9.1 How GeoCensus works

By using GeoCensus, the USA Regulatory Address Cleanse transform can append latitude, longitude, and census codes such as census tract and Metropolitan Statistical Area (MSA) to your records, based on ZIP+4 codes. MSA is an aggregation of US counties into Metropolitan Statistical Areas assigned by the US Office of Management and Budget. You can apply the GeoCensus codes during address standardization and postcode2 assignment for simple, “one-pass” processing.

The transform cannot, by itself, append demographic data to your records. The transform lays the foundation by giving you census coordinates via output fields. To append demographic information, you need a demographic database from another vendor. When you obtain one, we suggest that you use the matching process to match your records to the demographic database, and transfer the demographic information into your records. (You would use the MSA and census tract information as match criteria, then use the Best Record transform to post income and other information.)

Likewise, the transform does not draw maps. However, you can use the latitude and longitude assigned by the transform as input to third-party mapping applications. Those applications enable you to plot the locations of your customers and filter your database to cover a particular geographic area.

2.16.6.9.2 The software provides census coordinates

The software cannot, by itself, append demographic data to your records. The software simply lays the foundation by giving you census coordinates. To append demographic information, you need a demographic database from another vendor. When you get that, we suggest that you use our Match transform to match your records to the demographic database and transfer the demographic information into your records. (In technical terms, you would use the MSA and Census block/tract information as match fields, then use the Best Record post-match operation in the Match transform to transfer income and other information.

Likewise, the software does not draw maps. However, you can use the latitude and longitude assigned by the software as input to third-party mapping software. Those programs enable you to plot the locations of your customers and filter your database to cover a particular geographic area.

Related Information

Best record [page 562]

2.16.6.9.3 Get the most from the GeoCensus data

You can combine GeoCensus with the functionality of mapping software to view your geo-enhanced information. It will help your organization build its sales and marketing strategies. Here are some of the ways you can use the GeoCensus data, with or without mapping products.

Information type	How GeoCensus can help
Market analysis	You can use mapping applications to analyze market penetration, for instance. Companies striving to gain a clearer understanding of their markets employ market analysis. This way they can view sales, marketing, and demographic data on maps, charts, and graphs. The result is a more finely targeted marketing program. You will understand both where your customers are and the penetration you have achieved in your chosen markets.
Predictive modeling and target marketing	You can more accurately target your customers for direct response campaigns using geographic selections. Predictive modeling or other analytical techniques allow you to identify the characteristics of your ideal customer. This method incorporates demographic information used to enrich your customer database. From this analysis, it is possible to identify the best prospects for mailing or telemarketing programs.

Information type	How GeoCensus can help
Media planning	For better support of your advertising decisions, you may want to employ media planning. Coupling a visual display of key markets with a view of media outlets can help your organization make more strategic use of your advertising dollars.
Territory management	GeoCensus data provides a more accurate market picture for your organization. It can help you distribute territories and sales quotas more equitably.
Direct sales	Using GeoCensus data with market analysis tools and mapping software, you can track sales leads gathered from marketing activities.

2.16.6.9.4 GeoCensus directories

The path and file names for the following directories must be defined in the Reference Files option group of the USA Regulatory Address Cleanse transform before you can begin GeoCensus processing. You can use the substitution variable $$RefFilesDataCleanse.

Directory name	Description
ageo1-10	Address-level GeoCensus directories are required if you choose *Address* for the *Geo Mode* option under the Assignment Options group.
cgeo2.dir	Centriod-level GeoCensus directory is required if you choose *Centroid* for the *Geo Mode* option under the Assignment Options group.

2.16.6.9.5 GeoCensus mode options

To activate GeoCensus in the transform, you need to choose a mode in the *Geo Mode* option in the Assignment Options group.

Mode	Description
Address	Processes Address-level GeoCensus only.
Both	Attempts to make an Address-level GeoCensus assignment first. If no assignment is made, it attempts to make a Centroid-level GeoCensus assignment.
Centroid	Processes Centroid-level GeoCensus only.
None	Turns off GeoCensus processing.

2.16.6.9.6 GeoCensus output fields

You must include at least one of the following generated output fields in the USA Regulatory Address Cleanse transform if you plan to use the GeoCensus option:

- AGeo_CountyCode
- AGeo_Latitude
- AGeo_Longitude
- AGeo_MCDCode
- AGeo_PlaceCode
- AGeo_SectionCode
- AGeo_StateCode
- CGeo_BSACode
- CGeo_Latitude
- CGeo_Longitude
- CGeo_Metrocode
- CGeo_SectionCode

Find descriptions of these fields in the *Reference Guide*.

2.16.6.9.7 Sample transform configuration

To process with the GeoCensus feature in the USA Regulatory Address Cleanse transform, it is best to start with the sample transform configuration created for GeoCensus. Find the sample configuration, USARegulatoryGeo_AddressCleanse, under USA_Regulatory_Address_Cleanse in the Object Library.

2.16.6.9.8 To enable GeoCensus coding

If you use a copy of the USARegulatoryGeo_AddressCleanse sample transform file in your data flow, GeoCensus is already enabled. However, if you are starting from a USA Regulatory Address Cleanse transform, make sure you define the directory location and define the *Geo Mode* option.

1. Open the *USA Regulatory Address Cleanse* transform.
2. In the *Options* tab, expand the *Reference Files* group.
3. Set the locations for the `cgeo.dir` and `ageo1-10.dir` directories based on the Geo Mode you choose.
4. Expand the Assignment Options group, and select either *Address*, *Centroid*, or *Both* for the *Geo Mode* option.

   If you select *None*, the transform will not perform GeoCensus processing.

Related Information

GeoCensus (USA Regulatory Address Cleanse) [page 671]

2.16.6.10 Z4Change (USA Regulatory Address Cleanse)

The Z4Change option is based on a USPS directory of the same name. The Z4Change option is available in the USA Regulatory Address Cleanse transform only.

2.16.6.10.1 Use Z4Change to save time

Using the Z4Change option can save a lot of processing time, compared with running all records through the normal ZIP+4 assignment process.

Z4Change is most cost-effective for databases that are large and fairly stable—for example, databases of regular customers, subscribers, and so on. In our tests, based on files in which five percent of records were affected by a ZIP+4 change, total batch processing time was one third the normal processing time.

When you are using the transform interactively—that is, processing one address at a time—there is less benefit from using Z4Change.

2.16.6.10.2 USPS rules

Z4Change is to be used only for updating a database that has previously been put through a full validation process. The USPS requires that the mailing list be put through a complete assignment process every three years.

2.16.6.10.3 Z4Change directory

The Z4Change directory, `z4change.dir`, is updated monthly and is available only if you have purchased the Z4Change option for the USA Regulatory Address Cleanse transform.

The Z4Change directory contains a list of all the ZIP Codes and ZIP+4 codes in the country.

2.16.6.10.4 Start with a sample transform

If you want to use the Z4Change feature in the USA Regulatory Address Cleanse transform, it is best to start with the sample transform, USARegulatoryZ4Change_AddressCleanse.

2.16.6.10.5 To enable Z4Change

If you use a copy of the Z4Change transform configuration file sample (USARegulatoryZ4Change_AddressCleanse) in your data flow, Z4Change is already enabled. However, if you are starting from a USA Regulatory Address Cleanse transform, make sure you define the directory location and define the *Z4Change Mode* option.

1. Open the *USA Regulatory Address Cleanse* transform.
2. On the *Options* tab, expand the Reference Files group.
3. Set the location for the `z4change.dir` directory in the *Z4Change Directory* option.
4. Expand Z4Change options group and select *Yes* for the *Enable Z4Change* option.
5. In the Z4Change option group, enter the month and year that the input records were most recently ZIP+4 updated in the *Last ZIP+4 Assign Date* option.

2.16.6.11 Suggestion lists overview

Suggestion List processing is used in transactional projects with the USA Regulatory Address Cleanse, Global Address Cleanse, and the Global Suggestion List transforms. Use suggestion lists to complete and populate addresses that have minimal data. Suggestion lists can offer suggestions for possible matches if an exact match is not found. This section is only about suggestion lists in the USA Regulatory Address Cleanse transform.

> Note
>
> Suggestion list processing is not available for batch processing. In addition, if you have suggestion lists enabled, you are not eligible for CASS discounts and the software will not produce the required CASS documentation.

Related Information

Reference Guide: Global Address Cleanse suggestion lists [page 1257]
Integrator Guide: Using Data Services as a web service provider [page 2218]
Extracting data quality XML strings using extract_from_xml function [page 370]

2.16.6.11.1 Introduction to suggestion lists

Ideally, when the USA Regulatory Address Cleanse transform looks up an address in the USPS postal directories (City/ZCF), it finds exactly one matching record with a matching combination of locality, region, and postcode. Then, during the look-up in the USPS national ZIP+4 directory, the software should find exactly one record that matches the address.

Breaking ties

Sometimes it's impossible to pinpoint an inut address to one matching record in the directory. At other times, the software may find several directory records that are near matches to the input data.

When the software is close to a match, but not quite close enough, it assembles a list of the near matches and presents them as suggestions. When you choose a suggestion, the software tries again to assign the address.

Example

Incomplete last line

Given the incomplete last line below, the software could not reliably choose one of the four localities. But if you choose one, the software can proceed with the rest of the assignment process.

Input record	Possible matches in the City/ZCF directories

Line1= 1000 vine
Line2= lacr wi
	La Crosse, WI 54603
	Lancaster, WI 53813
	La Crosse, WI 54601
	Larson, WI 54947

Example

Missing directional

The same can happen with address lines. A common problem is a missing directional. In the example below, there is an equal chance that the directional could be North or South. The software has no basis for choosing one way or the other.

Input record	Possible matches in the ZIP+4 directory

Line1 = 615 losey blvd
Line2 = 54603
	600-699 Losey Blvd N
	600-698 Losey Blvd S

Example

Missing suffix

A missing suffix would cause similar behavior as in the example above.

Input record	Possible matches in the ZIP+4 directory

Line1 = 121 dorn
Line2 = 54601
	100-199 Dorn Pl
	101-199 Dorn St

Example

Misspelled street names

A misspelled or incomplete street name could also result in the need to be presented with address suggestions.

Input record	Possible matches in the ZIP+4 directory

Line1 = 4100 marl
Line2 = minneapolis mn
	4100-4199 Marshall 55421
	4100-4199 Maryland 55427

2.16.6.11.1.1 More information is needed

When the software produces a suggestion list, you need some basis for selecting one of the possible matches. Sometimes you need more information before choosing a suggestion.

Example

- Operators taking information over the phone can ask for more information from the customer to decide which suggestion list to choose.
- Operators entering data from a consumer coupon that is a little smudged may be able to choose a suggestion based on the information that is not smudged.

2.16.6.11.1.2 CASS rule

The USPS does not permit SAP Data Services to generate a USPS Form 3553 when suggestion lists are used in address assignment. The USPS suspects that users may be tempted to guess, which may result in misrouted mail that is expensive for the USPS to handle.

Therefore, when you use the suggestion list feature, you cannot get a USPS Form 3553 covering the addresses that you assign. The form is available only when you process in batch mode with the *Disable Certification* option set to No.

You must run addresses from real-time processes through a batch process in order to be CASS compliant. Then the software generates a USPS Form 3553 that covers your entire mailing database, and your list may be eligible for postal discounts.

2.16.6.11.1.3 Integrating functionality

Suggestion Lists functionality is designed to be integrated into your own custom applications via the Web Service. For information about integrating Data Services for web applications, see the *Integrator Guide*.

2.16.6.11.1.4 Sample suggestion lists blueprint

If you want to use the suggestion lists feature, it is best to start with one of the sample transforms configured for it. The sample transform is named USARegulatorySuggestions_Address_Cleanse. It is available for the USA Regulatory Address Cleanse transform.

2.16.6.12 Multiple data source statistics reporting

Statistics based on logical groups

For the USA Regulatory Address Cleanse transform, an input database can be a compilation of lists, with each list containing a field that includes a unique identifier. The unique identifier can be a name or a number, but it must reside in the same field across all lists.

The software collects statistics for each list using the Data_Source_ID input field. You map the field that contains the unique identifier in your list to the software's Data_Source_ID input field. When the software generates reports, some of the reports will contain a summary for the entire list, and a separate summary per list based on the value mapped into the Data_Source_ID field.

> ⚠ Restriction
>
> For compliance with NCOALink reporting restrictions, the USA Regulatory Address Cleanse transform does not support processing multiple mailing lists associated with different PAFs. Therefore, for NCOALink processing, all records in the input file are considered to be a single mailing list and are reported as such in the Customer Service Log (CSL) file.

> ⚠ Restriction
>
> The Gather Statistics Per Data Source functionality is not supported when the *Enable Parse Only* or *Enable Geo Only* options in the Non Certified Options group are set to Yes.

Related Information

Gathering statistics per list [page 680]

2.16.6.12.1 Data_Source_ID field

The software tracks statistics for each list based on the Data_Source_ID input field.

Example

In this example, there are five mailing lists combined into one list for input into the USA Regulatory Address Cleanse transform. Each list has a common field named List_ID, and a unique identifier in the List_ID field: N, S, E, W, C. The input mapping looks like this:

Transform input field name	Input schema column name	Type
DATA_SOURCE_ID	LIST_ID	varchar(80)

To obtain DPV statistics for each List_ID, process the job and then open the US Addressing report.

The first DPV Summary section in the US Addressing report lists the Cumulative Summary, which reports the totals for the entire input set. Subsequent DPV Summary sections list summaries per Data_Source_ID. The example in the table below shows the counts and percentages for the entire database (cumulative summary) and for Data_Source_ID “N”.

Statistic	DPV cumulative summary count	%	DPV summary for Data_Source_ID “N”	%
DPV Validated Addresses	1,968	3.94	214	4.28
Addresses Not DPV Valid	3,032	6.06	286	5.72
CMRA Validated Addresses	3	0.01	0	0.00
DPV Vacant Addresses	109	0.22	10	0.20
DPV NoStats Addresses	162	0.32	17	0.34

Related Information

Group statistics reporting [page 683]

2.16.6.12.2 Gathering statistics per list

Before setting up the USA Regulatory Address Cleanse transform to gather statistics per list, identify the field that uniquely identifies each list. For example, a mailing list that is comprised of more than one source might contain lists that have a field named LIST_ID that uniquely identifies each list.

1. Open the USA Regulatory Address Cleanse transform in the data flow and then click the *Options* tab.
2. Expand the Report and Analysis group and select Yes for the *Generate Report Data* and the *Gather Statistics Per Data Source* options.
3. Click the *Input* tab and click the *Input Schema Column Name* field next to the Data_Source_ID field for uniquely identifying a list.
   A drop menu appears.
4. Click the drop menu and select the input field from your database that you've chosen as the common field for uniquely identifying a list. In the scenario above, that would be the LIST_ID field.

5. Continue with the remaining job setup tasks and execute your job.

2.16.6.12.3 Physical Source Field and Cumulative Summary

Some reports include a report per list based on the Data_Source_ID field (identified in the report footer by “Physical Source Field”), and a summary of the entire list (identified in the report footer by “Cumulative Summary”). However, the Address Standardization, Address Information Code, and USA Regulatory Locking reports do not include a Cumulative Summary. The records in these reports are sorted by the Data Source ID value.

Note

When you enable NCOALink, the software reports a summary per list only for the following sections of the NCOALink Processing Summary Report:

- NCOALink Move Type Summary
- NCOALink Return Code Summary
- ANKLink Return Code Summary

Special circumstances

There are some circumstances when the words “Cumulative Summary” and“ Physical Source Field” will not appear in the report footer sections.

- When the *Gather Statistics Per Data Source* option is set to No
- When the *Gather Statistics Per Data Source* option is set to Yes and there is only one Data Source ID value present in the list but it is empty

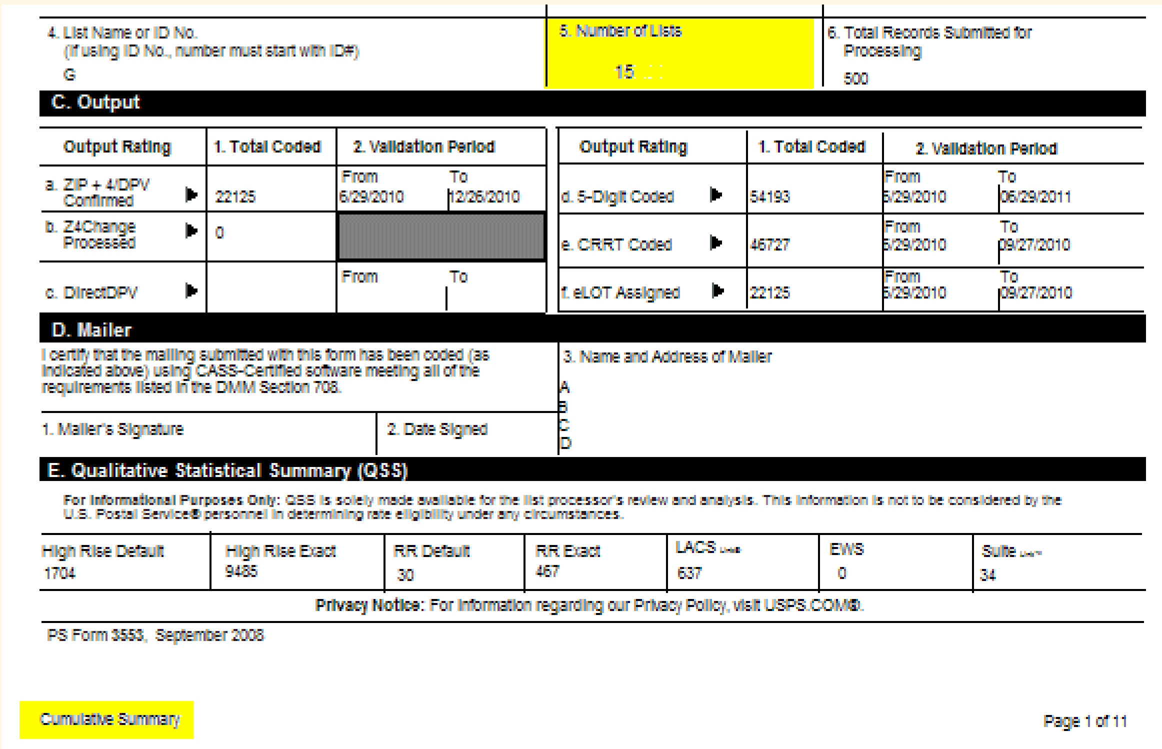
2.16.6.12.3.1 USPS Form 3553 and group reporting

The USPS Form 3553 includes a summary of the entire list and a report per list based on the Data_Source_ID field.

Example

Cumulative Summary

The USPS Form 3553 designates the summary for the entire list with the words “Cumulative Summary”. It appears in the footer as highlighted in the Cumulative Summary report sample below. In addition, the Cumulative Summary of the USPS Form 3553 contains the total number of lists in the job in Section B, field number 5, *Number of Lists* (highlighted below).



4. List Name or ID No. (If using ID No., number must start with ID#) G	5. Number of Lists 15	6. Total Records Submitted for Processing 500

C. Output

Output Rating	1. Total Coded	2. Validation Period		Output Rating	1. Total Coded	2. Validation Period	
		From	To			From	To
a. ZIP + 4/DPV Confirmed ▶	22125	6/29/2010	12/26/2010	d. 5-Digit Coded ▶	54193	6/29/2010	06/29/2011
b. Z4Change Processed ▶	0	▨	▨	e. CRRT Coded ▶	46727	From 6/29/2010	To 09/27/2010
c. DirectDPV ▶		From	To	f. eLOT Assigned ▶	22125	From 6/29/2010	To 09/27/2010

D. Mailer

I certify that the mailing submitted with this form has been coded (as indicated above) using CASS-Certified software meeting all of the requirements listed in the DMM Section 708.

1. Mailer's Signature

2. Date Signed

3. Name and Address of Mailer
A
B
C
D

E. Qualitative Statistical Summary (QSS)

For Informational Purposes Only: QSS is solely made available for the list processor's review and analysis. This information is not to be considered by the U.S. Postal Service® personnel in determining rate eligibility under any circumstances.

High Rise Default	High Rise Exact	RR Default	RR Exact	LACS LINK	EWS	Suite LINK
1704	9485	30	467	637	0	34

Privacy Notice: For information regarding our Privacy Policy, visit USPS.COM®.

PS Form 3553, September 2008

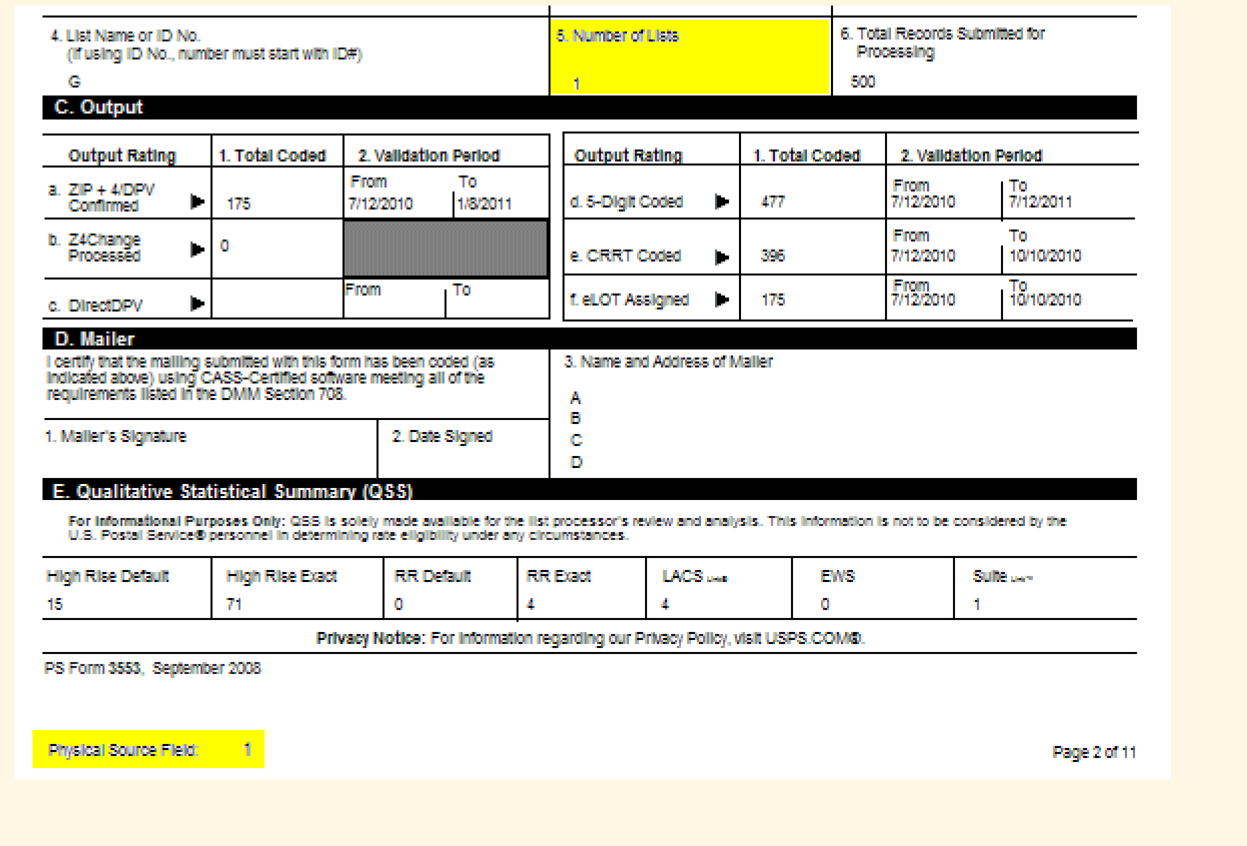
Cumulative Summary

Page 1 of 11

Example

Physical Source Field

The USPS Form 3553 designates the summary for each individual list with the words *Physical Source Field* followed by the Data Source ID value. It appears in the footer as highlighted in the sample below. The data in the report is for that list only.



4. List Name or ID No. (If using ID No., number must start with ID#) G	5. Number of Lists 1	6. Total Records Submitted for Processing 500

C. Output

Output Rating		1. Total Coded	2. Validation Period	
a. ZIP + 4/DPV Confirmed	▶	175	From 7/12/2010	To 1/8/2011
b. Z4Change Processed	▶	0		
c. DirectDPV	▶		From	To

Output Rating		1. Total Coded	2. Validation Period	
d. 5-Digit Coded	▶	477	From 7/12/2010	To 7/12/2011
e. CRRT Coded	▶	396	From 7/12/2010	To 10/10/2010
f. eLOT Assigned	▶	175	From 7/12/2010	To 10/10/2010

D. Mailer

I certify that the mailing submitted with this form has been coded (as indicated above) using CASS-Certified software meeting all of the requirements listed in the DMM Section 708.

1. Mailer's Signature

2. Date Signed

3. Name and Address of Mailer

A
B
C
D

E. Qualitative Statistical Summary (QSS)

For Informational Purposes Only: QSS is solely made available for the list processor's review and analysis. This information is not to be considered by the U.S. Postal Service® personnel in determining rate eligibility under any circumstances.

High Rise Default	High Rise Exact	RR Default	RR Exact	LACS Link	EWS	Suite Link
15	71	0	4	4	0	1

Privacy Notice: For information regarding our Privacy Policy, visit USPS.COM®.

PS Form 3553, September 2008

Physical Source Field: 1

Page 2 of 11

2.16.6.12.3.2 Group statistics reports

Reports that show both cumulative statistics (summaries for the entire mailing list) and group statistics (based on the Physical Source Field) include the following reports:

- Address Validation Summary
- Address Type Summary
- US Addressing

Reports that do not include a Cumulative Summary include the following:

- Address Information Code Summary
- Address Standardization
- US Regulatory Locking

Related Information

Sample data from US Addressing Report [page 679]

2.16.7 Data Quality support for native data types

The Data Quality transforms generally process incoming data types as character data. Therefore, if a noncharacter data type is mapped as input, the software converts the data to a character string before passing it through the Data Quality transforms.

Some Data Quality data types are recognized and processed as the same data type as they were input. For example, if a date type field is mapped to a Data Quality date type input field, the software has the following advantages:

- Sortation: The transform recognizes and sorts the incoming data as the specified data type.
- Efficiency: The amount of data being converted to character data is reduced making processing more efficient.

Related Information

Reference Guide: Data Quality transforms [page 1123]
Reference Guide: Data types [page 1027]

2.16.7.1 Data Quality data type definitions

The Data Quality transforms have four field attributes to define the field:

- Name
- Type
- Length
- Scale

These attributes are listed in the Input and output tab of the transform editor.

In the Input tab, the attribute Name is listed under the *Transform Input Field Name* column. The Type, Length, and Scale attributes are listed under the *Type* column in the format <type>(<length>, <scale>).

The Output tab also contains the four field attributes listed above. The attribute Name is listed under the *Field_Name* column. The Type, Length, and Scale attributes are listed under the *Type* column in the format <type>(<length>, <scale>).

2.16.8 Data Quality support for NULL values

The Data Quality transforms process NULL values as NULL.

A field that is NULL is passed through processing with the NULL marker preserved unless there is data available to populate the field on output. When there is data available, the field is output with the data available instead of NULL. The benefit of this treatment of NULL is that the Data Quality transforms treat a NULL marker as unknown instead of empty.

ℹ Note

If all fields of a record contain NULL, the transform will not process the record, and the record will not be a part of statistics and reports.

Related Information

Reference Guide: Data Quality transforms [page 1123]
Reference Guide: NULL values and empty strings [page 1715]

2.17 Design and Debug

This section covers the following Designer features that you can use to design and debug jobs:

- Use the View Where Used feature to determine the impact of editing a metadata object (for example, at table). See which data flows use the same object.
- Use the View Data feature to view sample source, transform, and target data in a data flow after a job executes.
- Use the Design-Time Data Viewer feature to view and analyze the input and output for a data set in real time as you design a transform.
- Use the Interactive Debugger to set breakpoints and filters between transforms within a data flow and view job data row-by-row during a job execution.
- Use the Difference Viewer to compare the metadata for similar objects and their properties.
- Use the auditing data flow feature to verify that correct data is processed by a source, transform, or target object.

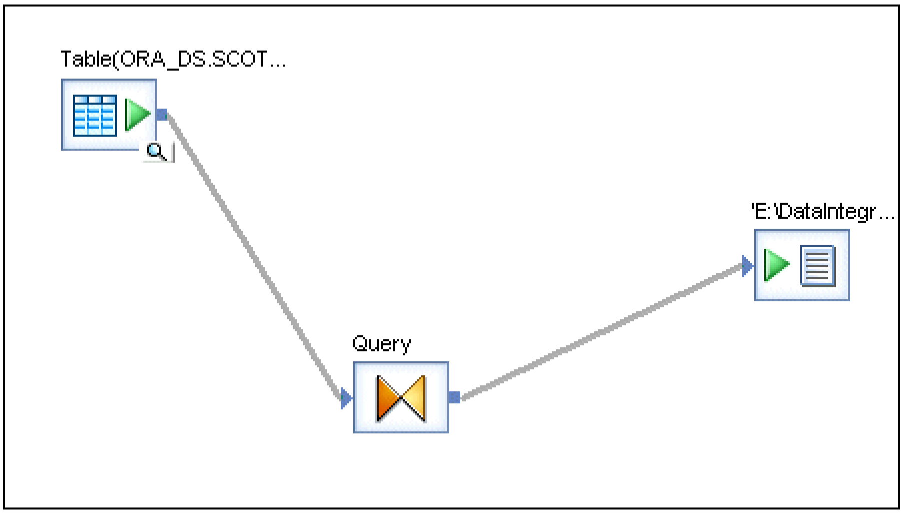
Related Information

Using View Where Used [page 686]
Using View Data [page 688]
Using the Design-Time Data Viewer [page 699]
Using the interactive debugger [page 701]
Comparing Objects [page 714]
Using Auditing [page 452]

2.17.1 Using View Where Used

When you save a job, work flow, or data flow the software also saves the list of objects used in them in your repository. Parent/child relationship data is preserved. For example, when the following parent data flow is saved, the software also saves pointers between it and its three children:

- a table source
- a query transform
- a file target



You can use this parent/child relationship data to determine what impact a table change, for example, will have on other data flows that are using the same table. The data can be accessed using the *View Where Used* option.

For example, while maintaining a data flow, you may need to delete a source table definition and re-import the table (or edit the table schema). Before doing this, find all the data flows that are also using the table and update them as needed.

To access the *View Where Used* option in the Designer you can work from the object library or the workspace.

2.17.1.1 Accessing View Where Used from the object library

You can view how many times an object is used and then view where it is used.

2.17.1.1.1 To access parent/child relationship information from the object library

1. View an object in the object library to see the number of times that it has been used.

   The Usage column is displayed on all object library tabs except:

   ○ Projects

- Jobs
- Transforms

Click the Usage column heading to sort values. For example, to find objects that are not used.

2. If the *Usage* is greater than zero, right-click the object and select *View Where Used*.

   The *Output* window opens. The Information tab displays rows for each parent of the object you selected. The type and name of the selected object is displayed in the first column's heading.

   The As column provides additional context. The As column tells you how the selected object is used by the parent.

   Other possible values for the As column are:

   - For XML files and messages, tables, flat files, etc., the values can be Source or Target
   - For flat files and tables only:

As	Description
Lookup()	Lookup table/file used in a `lookup` function
Lookup_ext()	Lookup table/file used in a `lookup_ext` function
Lookup_seq()	Lookup table/file used in a `lookup_seq` function

   - For tables only:

As	Description
Comparison	Table used in the Table Comparison transform
Key Generation	Table used in the Key Generation transform

3. From the *Output* window, double-click a parent object.

   The workspace diagram opens highlighting the child object the parent is using.

   Once a parent is open in the workspace, you can double-click a row in the output window again.

   - If the row represents a different parent, the workspace diagram for that object opens.
   - If the row represents a child object in the same parent, this object is simply highlighted in the open diagram.
     This is an important option because a child object in the *Output* window might not match the name used in its parent. You can customize workspace object names for sources and targets.
     The software saves both the name used in each parent and the name used in the object library. The Information tab on the *Output* window displays the name used in the object library. The names of objects used in parents can only be seen by opening the parent in the workspace.

2.17.1.2 Accessing View Where Used from the workspace

From an open diagram of an object in the workspace (such as a data flow), you can view where a parent or child object is used:

- To view information for the open (parent) object, select *View* > *Where Used*, or from the tool bar, select the *View Where Used* button.
  The *Output* window opens with a list of jobs (parent objects) that use the open data flow.
- To view information for a child object, right-click an object in the workspace diagram and select the *View Where Used* option.
  The *Output* window opens with a list of parent objects that use the selected object. For example, if you select a table, the *Output* window displays a list of data flows that use the table.

2.17.1.3 Limitations

- This feature is not supported in central repositories.
- Only parent and child pairs are shown in the *Information* tab of the Output window.
  For example, for a table, a data flow is the parent. If the table is also used by a grandparent (a work flow for example), these are not listed in the *Output* window display for a table. To see the relationship between a data flow and a work flow, open the work flow in the workspace, then right-click a data flow and select the *View Where Used* option.
- The software does not save parent/child relationships between functions.
  - If function `A` calls function `B`, and function `A` is not in any data flows or scripts, the *Usage* in the object library will be zero for both functions. The fact that function `B` is used once in function `A` is not counted.
  - If function `A` is saved in one data flow, the usage in the object library will be `1` for both functions `A` and `B`.
- Transforms are not supported. This includes custom ABAP transforms that you might create to support an SAP applications environment.
- The Designer counts an object's usage as the number of times it is used for a unique purpose. For example, in data flow `DF1` if table `DEPT` is used as a source twice and a target once the object library displays its *Usage* as `2`. This occurrence should be rare. For example, a table is not often joined to itself in a job design.

2.17.2 Using View Data

View Data provides a way to scan and capture a sample of the data produced by each step in a job, even when the job does not execute successfully. View imported source data, changed data from transformations, and ending data at your targets. At any point after you import a data source, you can check on the status of that data—before and after processing your data flows.

Use View Data to check the data while designing and testing jobs to ensure that your design returns the results you expect. Using one or more View Data panes, you can view and compare sample data from different steps. View Data information is displayed in embedded panels for easy navigation between your flows and the data.

Use View Data to look at:

- Sources and targets
  View Data allows you to see data before you execute a job. Armed with data details, you can create higher quality job designs. You can scan and analyze imported table and file data from the object library as well as see the data for those same objects within existing jobs. After you execute the job, you can refer back to the source data again.
- Transforms

- Lines in a diagram

ℹ Note

- View Data displays blob data as `<blob>`.
- View Data is not supported for SAP IDocs. For SAP and PeopleSoft, the Table Profile tab and Column Profile tab options are not supported for hierarchies.

Related Information

Viewing data passed by transforms [page 712]
Using the interactive debugger [page 701]

2.17.2.1 Accessing View Data

2.17.2.1.1 To view data for sources and targets

You can view data for sources and targets from two different locations:

1. *View Data button*

   View Data buttons appear on source and target objects when you drag them into the workspace. Click the View data button (magnifying glass icon) to open a View Data pane for that source or target object.
2. *Object library*

   View Data in potential source or target objects from the Datastores or Formats tabs.

   Open a View Data pane from the object library in one of the following ways:
   - Right-click a table object and select *View Data*.
   - Right-click a table and select *Open* or *Properties*.
     The Table Metadata, XML Format Editor, or Properties window opens. From any of these windows, you can select the View Data tab.

To view data for a file, the file must physically exist and be available from your computer's operating system. To view data for a table, the table must be from a supported database.

Related Information

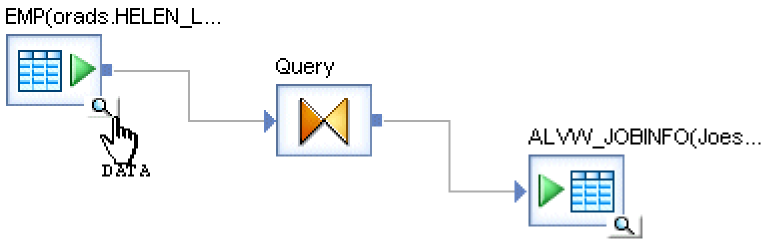
Viewing data in the workspace [page 690]

2.17.2.2 Viewing data in the workspace

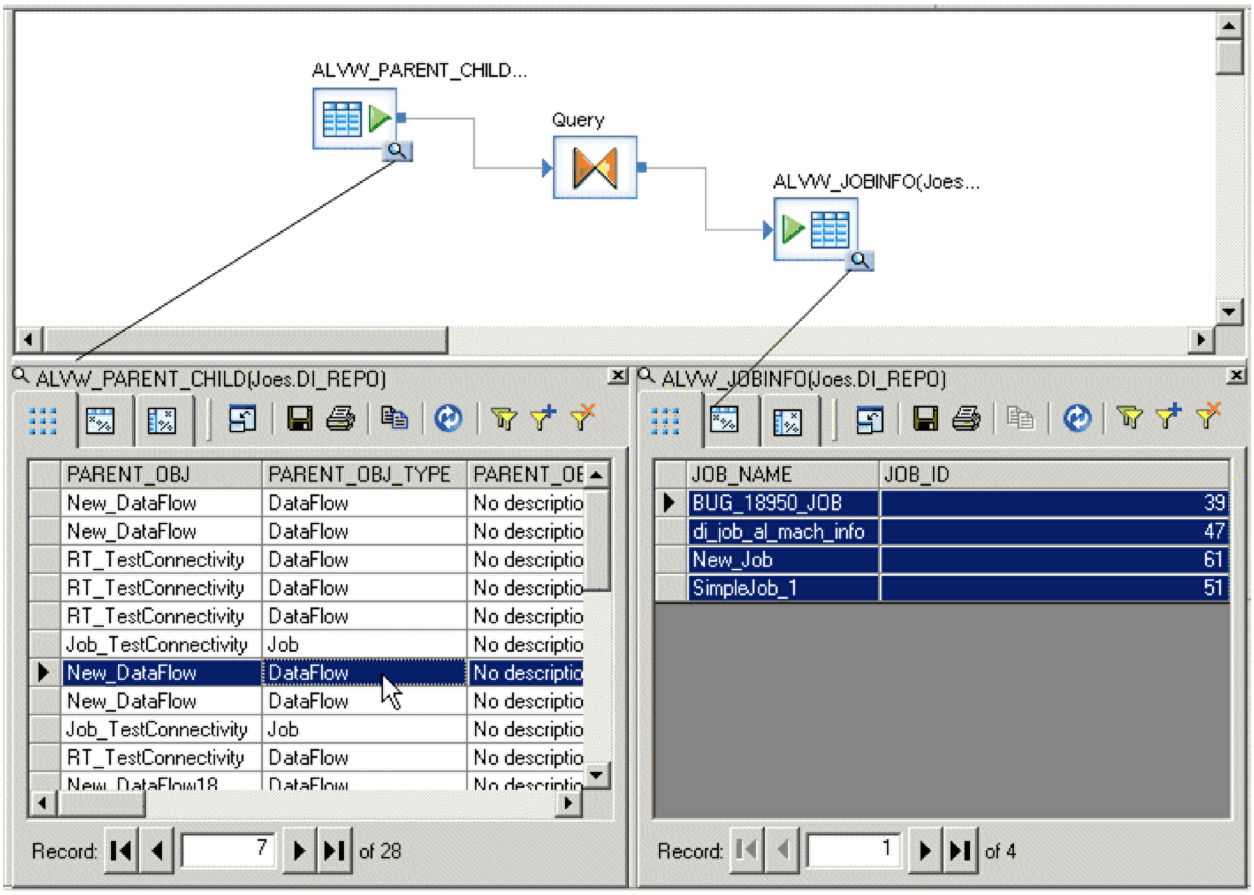
View Data can be accessed from the workspace when magnifying glass buttons appear over qualified objects in a data flow. This means:

For sources and targets, files must physically exist and be accessible from the Designer, and tables must be from a supported database.

To open a View Data pane in the Designer workspace, click the magnifying glass button on a data flow object.



A large View Data pane appears beneath the current workspace area. Click the magnifying glass button for another object and a second pane appears below the workspace area. (Note that the first pane area shrinks to accommodate the presence of the second pane).



ALVW_PARENT_CHILD(Joes.DI_REPO)

PARENT_OBJ	PARENT_OBJ_TYPE	PARENT_OE
New_DataFlow	DataFlow	No descriptio
New_DataFlow	DataFlow	No descriptio
RT_TestConnectivity	DataFlow	No descriptio
RT_TestConnectivity	DataFlow	No descriptio
RT_TestConnectivity	DataFlow	No descriptio
Job_TestConnectivity	Job	No descriptio
New_DataFlow	DataFlow	No descriptio
New_DataFlow	DataFlow	No descriptio
Job_TestConnectivity	Job	No descriptio
RT_TestConnectivity	DataFlow	No descriptio
New_DataFlow18	DataFlow	No descriptio

Record: 7 of 28

ALVW_JOBINFO(Joes.DI_REPO)

JOB_NAME	JOB_ID
BUG_18950_JOB	39
di_job_al_mach_info	47
New_Job	61
SimpleJob_1	51

Record: 1 of 4

You can open two View Data panes for simultaneous viewing. When both panes are filled and you click another View Data button, a small menu appears containing window placement icons. The black area in each icon indicates the pane you want to replace with a new set of data. Click a menu option and the data from the latest selected object replaces the data in the corresponding pane.

The description or path for the selected View Data button displays at the top of the pane.

- For sources and targets, the description is the full object name:
  - `<ObjectName>` (`<Datastore.Owner>`) for tables
  - `<FileName>` (`<File Format Name>`) for files
- For View Data buttons on a line, the path consists of the object name on the left, an arrow, and the object name to the right.
  For example, if you select a View Data button on the line between the query named Query and the target named ALVW_JOBINFO(joes.DI_REPO), the path would indicate:

```
Query -> ALVW_JOBINFO(Joes.DI_REPO)
```

You can also find the View Data pane that is associated with an object or line by:

- Rolling your cursor over a View Data button on an object or line. The Designer highlights the View Data pane for the object.

- Looking for grey View Data buttons on objects and lines. The Designer displays View Data buttons on open objects with grey rather than white backgrounds.

Related Information

Viewing data passed by transforms [page 712]

2.17.2.3 View Data Properties

You can access View Data properties from tool bar buttons or the right-click menu.

View Data displays your data in the rows and columns of a data grid. The number of rows displayed is determined by a combination of several conditions:

- Sample size — The number of rows sampled in memory. Default sample size is 1000 rows for imported source and target objects. Maximum sample size is 5000 rows. Set sample size for sources and targets from *Tools* > *Options* > *Designer* > *General* > *View Data sampling size*.
  When using the interactive debugger, the software uses the Data sample rate option instead of sample size.
- Filtering
- Sorting

If your original data set is smaller or if you use filters, the number of returned rows could be less than the default.

You can see which conditions have been applied in the *navigation bar*.

Related Information

Filtering [page 692]
Sorting [page 694]
Starting and stopping the interactive debugger [page 705]

2.17.2.3.1 Filtering

You can focus on different sets of rows in a local or new data sample by placing fetch conditions on columns.

2.17.2.3.1.1 To view and add filters

1. In the View Data tool bar, click the Filters button, or right-click the grid and select *Filters*.

The Filters window opens.

2. Create filters.

   The Filters window has three columns:

   a) Column—Select a name from the first column. Select *{remove filter}* to delete the filter.
   b) Operator—Select an operator from the second column.
   c) Value—Enter a value in the third column that uses one of the following data type formats

Data Type	Format
Integer, double, real	standard
date	yyyy.mm.dd
time	hh24:mm:ss
datetime	yyyy.mm.dd hh24:mm.ss
varchar	'abc'

3. In the *Concatenate all filters using* list box, select an operator (*AND*, *OR*) for the engine to use in concatenating filters.

   Each row in this window is considered a filter.

4. To see how the filter affects the current set of returned rows, click *Apply*.
5. To save filters and close the Filters window, click *OK*.

   Your filters are saved for the current object and the local sample updates to show the data filtered as specified in the Filters dialog. To use filters with a new sample, see *Using Refresh*.

Related Information

Using Refresh [page 694]

2.17.2.3.1.2 To add a filter for a selected cell

1. Select a cell from the sample data grid.
2. In the View Data tool bar, click the Add Filter button, or right-click the cell and select *Add Filter*.

   The Add Filter option adds the new filter condition, <column> = <cell value>, then opens the Filters window so you can view or edit the new filter.

3. When you are finished, click *OK*.

To remove filters from an object, go to the View Data tool bar and click the Remove Filters button, or right-click the grid and select *Remove Filters*. All filters are removed for the current object.

2.17.2.3.2 Sorting

You can click one or more column headings in the data grid to sort your data. An arrow appears on the heading to indicate sort order: ascending (up arrow) or descending (down arrow).

To change sort order, click the column heading again. The priority of a sort is from left to right on the grid.

To remove sorting for an object, from the tool bar click the Remove Sort button, or right-click the grid and select *Remove Sort*.

Related Information

Using Refresh [page 694]

2.17.2.3.3 Using Refresh

To fetch another data sample from the database using new filter and sort settings, use the *Refresh* command. After you edit filtering and sorting, in the tool bar click the Refresh button in the tool bar, or right-click the data grid and select *Refresh*.

To stop a refresh operation, click the Stop button. While the software is refreshing the data, all View Data controls except the *Stop* button are disabled.

2.17.2.3.4 Using Show/Hide Columns

You can limit the number of columns displayed in View Data by using the *Show/Hide Columns* option from:

- The tool bar.
- The right-click menu.
- The arrow shortcut menu, located to the right of the *Show/Hide Columns* tool bar button. This option is only available if the total number of columns in the table is ten or fewer. Select a column to display it.

You can also "quick hide" a column by right-clicking the column heading and selecting *Hide* from the menu.

2.17.2.3.4.1 To show or hide columns

1. Click the Show/Hide columns tool bar button, or right-click the data grid and select *Show/Hide Columns*.

   The Column Settings window opens.
2. Select the columns that you want to display or click one of the following buttons: *Show*, *Show All*, *Hide*, or *Hide All*.
3. Click *OK*.

2.17.2.3.5 Opening a new window

To see more of the data sample that you are viewing in a View Data pane, open a full-sized View Data window. From any View Data pane, click the Open Window tool bar button to activate a separate, full-sized View Data window. Alternatively, you can right-click and select *Open in new window* from the menu.

2.17.2.4 View Data tool bar options

The following options are available on View Data panes.

Icon	Option	Description
	Open in new window	Opens the View Data pane in a larger window.
	Save As	Saves the data in the View Data pane.
	Print	Prints View Data pane data.
	Copy Cell	Copies View Data pane cell data.
	Refresh data	Fetches another data sample from existing data in the View Data pane using new filter and sort settings.
	Open Filters window	Opens the Filters window.
	Add a Filter	Adds a filter to a selected cell.
	Remove Filter	Removes all filters in the View Data pane.
	Remove Sort	Removes sort settings for the object you select.

Icon	Option	Description
	Show/hide navigation	Shows or hides the navigation bar which appears below the data table.
	Show/hide columns	

Related Information

Opening a new window [page 695]
Using Refresh [page 694]
Filtering [page 692]
Sorting [page 694]
To add a filter for a selected cell [page 693]
Using Show/Hide Columns [page 694]

2.17.2.5 View Data tabs

The View Data panel for objects contains three tabs:

- Data tab
- Profile tab
- Column Profile tab

Use tab options to give you a complete profile of a source or target object. The Data tab is always available. The Profile and Relationship tabs are supported with the Data Profiler. Without the Data Profiler, the Profile and Column Profile tabs are supported for some sources and targets (see *Release Notes* for more information).

Related Information

Viewing the profiler results [page 443]

2.17.2.5.1 Data tab

The Data tab allows you to use the properties of *View Data*. It also indicates nested schemas such as those used in XML files and messages. When a column references nested schemas, that column is shaded yellow and a small table icon appears in the column heading.

Related Information

View Data Properties [page 692]

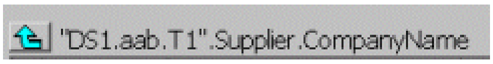
2.17.2.5.1.1 To view a nested schema

1. Double-click a cell.

   The data grid updates to show the data in the selected cell or nested table.

   In the Schema area, the selected cell value is marked by a special icon. Also, tables and columns in the selected path are displayed in blue, while nested schema references are displayed in grey.

   In the Data area, data is shown for columns. Nested schema references are shown in angle brackets; for example, **`<CompanyName>`**.
2. Continue to use the data grid side of the panel to navigate. For example:
   - Select a lower-level nested column and double-click a cell to update the data grid.
   - Click the at the top of the data grid to move up in the hierarchy.

     "DS1.aab.T1".Supplier.CompanyName

   - See the entire path to the selected column or table displayed to the right of the Drill Up button. Use the path and the data grid to navigate through nested schemas.

2.17.2.5.2 Profile tab

If you use the Data Profiler, the Profile tab displays the profile attributes that you selected on the *Submit Column Profile Request* option.

The Profile tab allows you to calculate statistical information for any set of columns you choose. This optional feature is not available for columns with nested schemas or for the LONG data type.

Related Information

Executing a profiler task [page 438]

2.17.2.5.2.1 To use the Profile tab without the Data Profiler

1. Select one or more columns.

   Select only the column names you need for this profiling operation because Update calculations impact performance.

   You can also right-click to use the Select All and Deselect All menu options.
2. Click *Update*.
3. The statistics appear in the Profile grid.

   The grid contains six columns:

Column	Description
Column	Names of columns in the current table. Select names from this column, then click *Update* to populate the profile grid.
Distinct Values	The total number of distinct values in this column.
NULLs	The total number of NULL values in this column.
Min	Of all values, the minimum value in this column.
Max	Of all values, the maximum value in this column.
Last Updated	The time that this statistic was calculated.

   Sort values in this grid by clicking the column headings. Note that Min and Max columns are not sortable.

In addition to updating statistics, you can click the *Records* button on the Profile tab to count the total number of physical records in the object you are profiling.

The software saves previously calculated values in the repository and displays them until the next update.

2.17.2.5.3 Column Profile tab

The Column Profile tab allows you to calculate statistical information for a single column. If you use the Data Profiler, the Relationship tab displays instead of the Column Profile.

> **Note**
>
> This optional feature is not available for columns with nested schemas or the LONG data type.

Related Information

To view the relationship profile data generated by the Data Profiler [page 446]

2.17.2.5.3.1 To calculate value usage statistics for a column

1. Enter a number in the *Top* box.

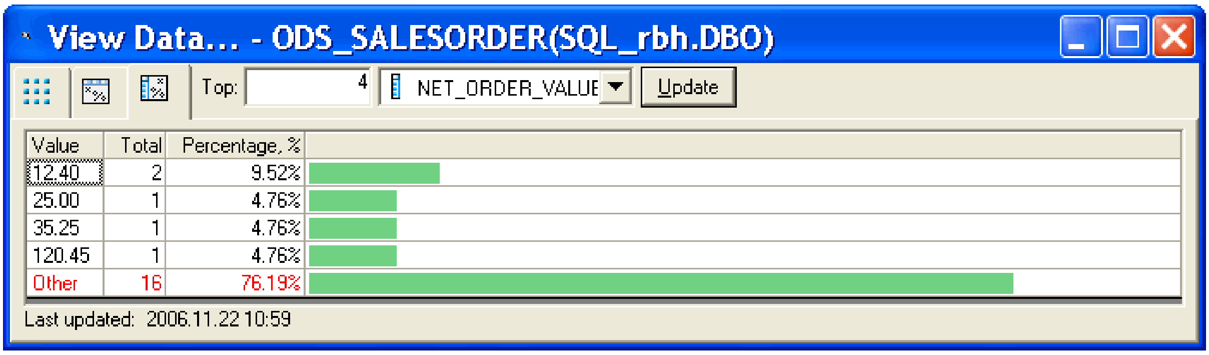
   This number is used to find the most frequently used values in the column. The default is 10, which means that the software returns the top 10 most frequently used values.
2. Select a column name in the list box.
3. Click *Update*.

   The Column Profile grid displays statistics for the specified column. The grid contains three columns:

Column	Description
Value	A top (most frequently used) value found in your specified column, or *Other* (remaining values that are not used as frequently).
Total	The total number of rows in the specified column that contain this value.
Percentage	The percentage of rows in the specified column that have this value compared to the total number of values in the column.

The software returns a number of values up to the number specified in the *Top* box plus an additional value called *Other*.

So if you enter 4 in the Top box, you will get up to five returned values (the top-four-used values in the specified column, plus the Other category). Results are saved in the repository and displayed until you perform a new update.



View Data... - ODS_SALESORDER(SQL_rbh.DBO)

Top: 4 NET_ORDER_VALUE Update

Value	Total	Percentage, %
12.40	2	9.52%
25.00	1	4.76%
35.25	1	4.76%
120.45	1	4.76%
Other	16	76.19%

Last updated: 2006.11.22 10:59

For example, statistical results in the preceding table indicate that of the four most frequently used values in the NET_ORDR_VALUE column, 9.52% use the value 12.40, 4.76% use the value 25.00, and so on. You can also see that the four most frequently used values (the "top four") are used in approximately 24% of all cases because approximately 76% is shown in the Other category. For this example, the total number of rows counted during the calculation for each top value is 21.

2.17.3 Using the Design-Time Data Viewer

The Design-Time Data Viewer lets you view and analyze the input and output for a data set in real time as you design a transform. The data flow does not need to be complete or valid, although it must use a valid, accessible source that contains data.

Use the Design-Time Data Viewer to view the data while designing a transform to ensure that your design returns the results that you expect. The Design-Time Data Viewer displays as input and output panes in the transform editor so that you can compare the data before and after the transform acts on it.

To use the Design-Time Data Viewer for the Global Address Cleanse, Geocoder, and USA Regulatory Address Cleanse transforms, you must have access to the Data Quality reference data (directories).

Related Information

Viewing Design-Time Data [page 700]
Configuring the Design-Time Data Viewer [page 700]

2.17.3.1 Viewing Design-Time Data

1. To view the input and output data for transforms in the transform editor, select *Debug* > *View Design-Time Data*.

   Input and output data panes open in the transform editor. Each pane may contain several tabbed views depending on how many inputs and outputs the selected transform has. For example, although most transforms have only one output, the Validation transform has three outputs.
2. To view the data for a different input or output, click the appropriate tab in the View Data pane.
3. To view the input and output data automatically after you modify a transform, select *Debug* > *View Automatically*.
4. To filter the number of data rows displayed in the panes, select *Debug* > *Filter Input Dataset*. Setting the filter may help increase performance while you are designing the transform and have the Design-Time Data Viewer feature set to update automatically.

   By default, the filter displays the first 50 rows. You can configure the number of rows that are displayed in the *Options* window.
5. To close the Design-Time Data Viewer panes, click the x in the upper-right corner of the pane.

Related Information

Configuring the Design-Time Data Viewer [page 700]

2.17.3.2 Configuring the Design-Time Data Viewer

You can configure the number of data rows that are displayed in the Design-Time Data Viewer panes as well as the time that is allowed for updates before it times out.

1. To configure the Design-Time Data Viewer options, select *Debug* › *Options*.
   A window opens and displays the available options.
2. Edit the options as necessary.

Option	Description
Number of rows to read from any source when filtered	Specifies the number of rows that are read from the source if the Filter Input Dataset menu item is selected. By default, the filter reads the first 50 rows of data from the source.
Time-out interval for automatic mode	Specifies the amount of time, in seconds, allowed to update the data if the View Automatically menu item is selected before timing out and returning an error.
Time-out interval for manual mode	Specifies the amount of time, in seconds, allowed to update the data if the View Automatically menu item is deselected before timing out and returning an error.

3. Click *OK* to close the window.

2.17.3.3 Specifying variables for expressions

You can specify variables for use in transformation expressions. The value of the variable is used in design-time data calculations. You can also set values for global variables that are used only for design-time calculations.

1. To configure the Design-Time Data Viewer options, select *Debug* › *Options*.
   A window opens and displays the available options.
2. In the Variables area, enter the name, data type, and value for the variable.
3. To import a global variable, click the *Import* button. All global variables from each job in the repository populate the table.
4. Enter the value for each imported global value.
   Variable values set in the Design-Time Data Viewer options are only used for design-time data calculations.
5. To remove a variable from the table, select it and click the *Delete* button.
6. Click *OK* to close the window.

2.17.4 Using the interactive debugger

The Designer includes an interactive debugger that allows you to examine and modify data row-by-row (during a debug mode job execution) by placing filters and breakpoints on lines in a data flow diagram. The interactive debugger provides powerful options to debug a job.

> Note
>
> A repository upgrade is required to use this feature.

2.17.4.1 Before starting the interactive debugger

Like executing a job, you can start the interactive debugger from the *Debug* menu when a job is active in the workspace. Select *Start debug*, set properties for the execution, then click *OK*. The debug mode begins. The Debug mode provides the interactive debugger's windows, menus, and tool bar buttons that you can use to control the pace of the job and view data by pausing the job execution using filters and breakpoints.

While in debug mode, all other Designer features are set to read-only. To exit the debug mode and return other Designer features to read/write, click the *Stop debug* button on the interactive debugger toolbar.

All interactive debugger commands are listed in the Designer's *Debug* menu. The Designer enables the appropriate commands as you progress through an interactive debugging session.

Before you start a debugging session, however, you might want to set the following:

- Filters and breakpoints
- Interactive debugger port between the Designer and an engine.

2.17.4.1.1 Setting filters and breakpoints

You can set any combination of filters and breakpoints in a data flow before you start the interactive debugger. The debugger uses the filters and pauses at the breakpoints you set.

If you do not set predefined filters or breakpoints:

- The Designer will optimize the debug job execution. This often means that the first transform in each data flow of a job is pushed down to the source database. Consequently, you cannot view the data in a job between its source and the first transform unless you set a predefined breakpoint on that line.
- You can pause a job manually by using a debug option called *Pause Debug* (the job pauses before it encounters the next transform).

Related Information

Push-down optimizer [page 713]

2.17.4.1.1.1 To set a filter or breakpoint

1. In the workspace, open the job that you want to debug.
2. Open one of its data flows.
3. Right-click the line that you want to examine and select *Set Filter/Breakpoint*.

   A line is a line between two objects in a workspace diagram.

   The Breakpoint window opens. Its title bar displays the objects to which the line connects.
4. Set and enable a filter or a breakpoint using the options in this window.

A debug filter functions as a simple Query transform with a WHERE clause. Use a filter to reduce a data set in a debug job execution. Note that complex expressions are not supported in a debug filter.

Place a debug filter on a line between a source and a transform or two transforms. If you set a filter and a breakpoint on the same line, The software applies the filter first. The breakpoint can only see the filtered rows.

Like a filter, you can set a breakpoint between a source and transform or two transforms. A breakpoint is the location where a debug job execution pauses and returns control to you.

Choose to use a breakpoint with or without conditions.

- If you use a breakpoint without a condition, the job execution pauses for the first row passed to a breakpoint.
- If you use a breakpoint with a condition, the job execution pauses for the first row passed to the breakpoint that meets the condition.

A breakpoint condition applies to the after image for UPDATE, NORMAL and INSERT row types and to the before image for a DELETE row type.

Instead of selecting a conditional or unconditional breakpoint, you can also use the Break after 'n' row(s) option. In this case, the execution pauses when the number of rows you specify pass through the breakpoint.

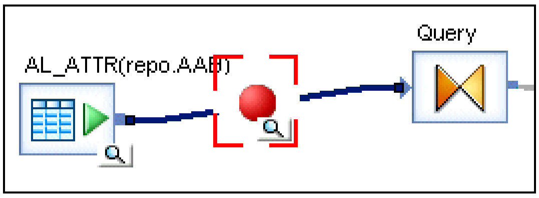
5. Click *OK*.

The Breakpoint enabled icon appears on the selected line.

The software provides the following filter and breakpoint conditions:

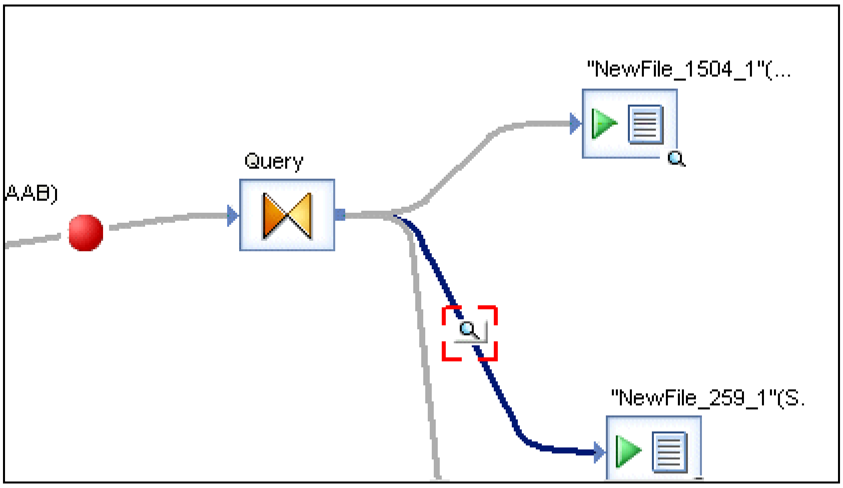
Icon	Description
⚪	Breakpoint disabled
🔴	Breakpoint enabled
▽	Filter disabled
▼	Filter enabled
▽⚪	Filter and breakpoint disabled
▼🔴	Filter and breakpoint enabled
▼⚪	Filter enabled and breakpoint disabled
▽🔴	Filter disabled and breakpoint enabled

In addition to the filter and breakpoint icons that can appear on a line, the debugger highlights a line when it pauses there. A red locator box also indicates your current location in the data flow. For example, when you start the interactive debugger, the job pauses at your breakpoint. The locator box appears over the breakpoint icon as shown in the following diagram:



A View Data button also appears over the breakpoint. You can use this button to open and close the View Data panes.

As the debugger steps though your job's data flow logic, it highlights subsequent lines and displays the locator box at your current position.



Related Information

Panes [page 706]

2.17.4.1.2 Changing the interactive debugger port

The Designer uses a port to an engine to start and stop the interactive debugger. The interactive debugger port is set to `5001` by default.

2.17.4.1.2.1 To change the interactive debugger port setting

1. Select *Tools* > *Options* > *Designer* > *Environment* .

2. Enter a value in the *Interactive Debugger* box.
3. Click *OK*.

2.17.4.2 Starting and stopping the interactive debugger

A job must be active in the workspace before you can start the interactive debugger. You can select a job from the object library or from the project area to activate it in the workspace. Once a job is active, the Designer enables the *Start Debug* option on the *Debug* menu and tool bar.

2.17.4.2.1 To start the interactive debugger

1. In the project area, right-click a job and select *Start debug*.

   Alternatively, in the project area you can click a job and then:

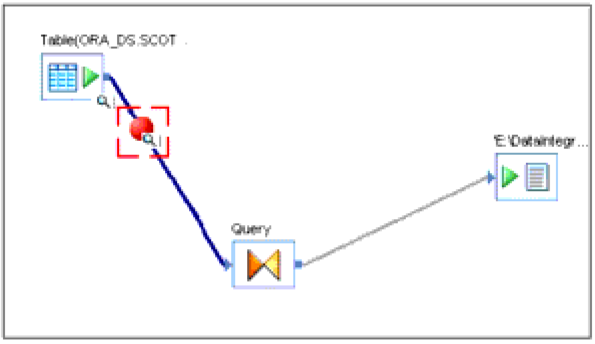
   - Press `Ctrl`+`F8`.
   - From the *Debug* menu, click *Start debug*.
   - Click the *Start debug* button on the tool bar.

   The Debug Properties window opens. The Debug Properties window includes three parameters similar to the Execution Properties window (used when you just want to run a job).

   You will also find more information about the Trace and Global Variable options.

   The options unique to the Debug Properties window are:

   - *Data sample rate*—The number of rows cached for each line when a job executes using the interactive debugger. For example, in the following data flow diagram, if the source table has 1000 rows and you set the *Data sample rate* to `500`, then the Designer displays up to 500 of the last rows that pass through a selected line. The debugger displays the last row processed when it reaches a breakpoint.

   - *Exit the debugger when the job is finished*—Click to stop the debugger and return to normal mode after the job executes. Defaults to cleared.
2. Enter the debug properties that you want to use or use the defaults.
3. Click *OK*.

   The job you selected from the project area starts to run in debug mode. The Designer:

- Displays the interactive debugger windows.
- Adds `Debugging Job` **`<JobName>`** to its title bar.
- Enables the appropriate *Debug* menu and tool bar options.
- Displays the debug icon in the status bar.
- Sets the user interface to read-only.

> i Note
>
> You cannot perform any operations that affect your repository (such as dropping objects into a data flow) when you execute a job in debug mode.

When the debugger encounters a breakpoint, it pauses the job execution. You now have control of the job execution. The interactive debugger windows display information about the job execution up to this point. They also update as you manually step through the job or allow the debugger to continue the execution.

Related Information

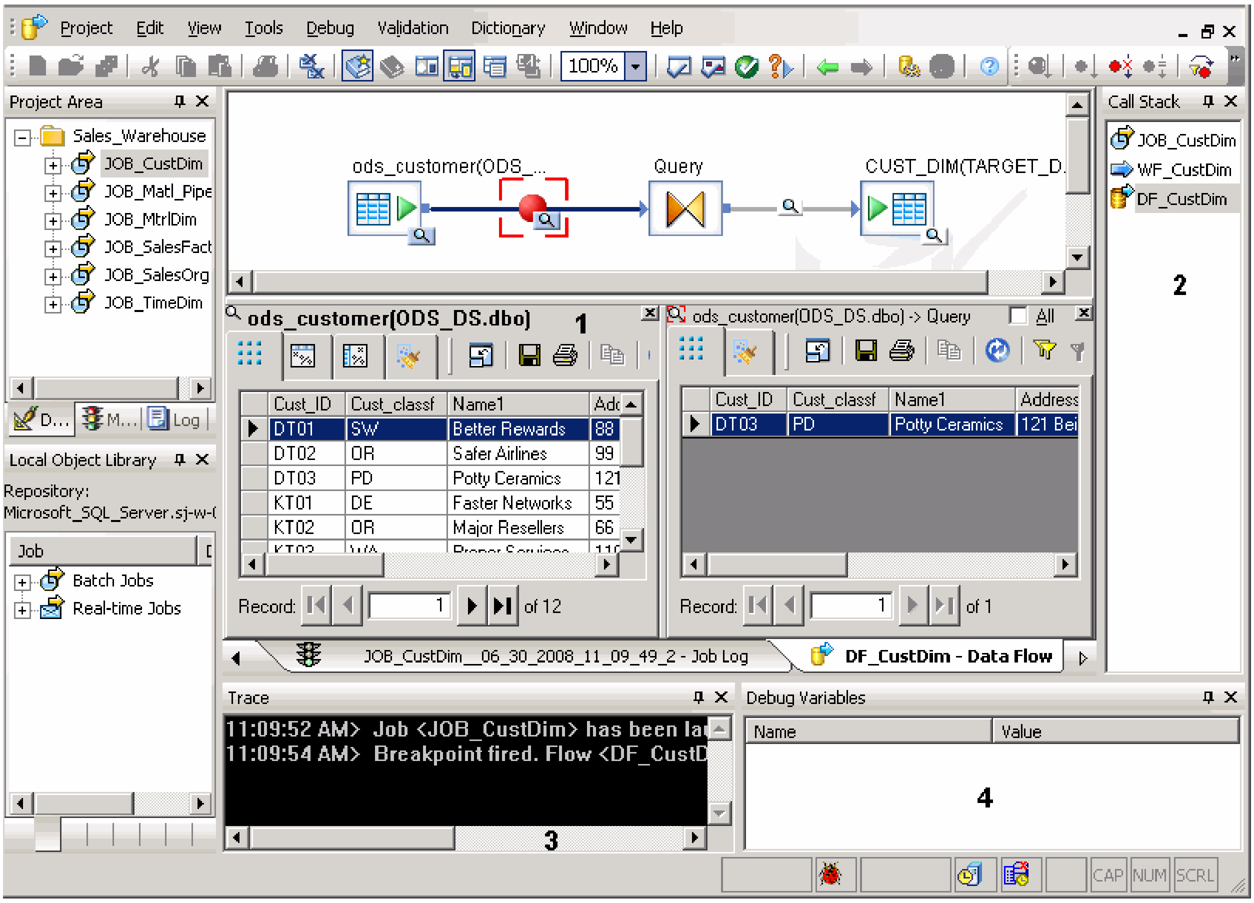
Reference Guide: Parameters [page 840]

2.17.4.2.2 To stop a job in debug mode and exit the interactive debugger

Click the *Stop Debug* button on the tool bar, press *Shift*+*F8*, or from the *Debug* menu, click *Stop debug*.

2.17.4.3 Panes

When you start a job in the interactive debugger, the Designer displays three additional panes as well as the View Data panes beneath the work space. The following diagram shows the default locations for these panes.



ods_customer(ODS_DS.dbo)

Cust_ID	Cust_classf	Name1	Add
DT01	SW	Better Rewards	88
DT02	OR	Safer Airlines	99
DT03	PD	Potty Ceramics	121
KT01	DE	Faster Networks	55
KT02	OR	Major Resellers	66
KT03	WA	Proper Services	11(

ods_customer(ODS_DS.dbo) -> Query

Cust_ID	Cust_classf	Name1	Address
DT03	PD	Potty Ceramics	121 Bei

Trace

```
11:09:52 AM> Job <JOB_CustDim> has been la
11:09:54 AM> Breakpoint fired. Flow <DF_CustD
```

1. View Data panes
2. Call Stack pane
3. Trace pane
4. Debug Variable pane

Each pane is docked in the Designer's window. To move a debugger pane, double-click its control bar to release it, then click and drag its title bar to re-dock it.

The Designer saves the layout you create when you stop the interactive debugger. Your layout is preserved for your next Designer session.

You can resize or hide a debugger pane using its control buttons. To show or hide a debugger pane manually, use the *Debug* menu or the tool bar.

Related Information

Debug menu options and tool bar [page 711]

2.17.4.3.1 Call stack window

The Call Stack window lists the objects in the path encountered so far (before either the job completes, encounters a breakpoint, or you pause it).

For example, for the job JOB_CustDim that includes a work flow and data flow, the Call Stack window might display:

JOB_CustDim

WF_CustDim

DF_CustDim

You can double-click an object in the Call Stack window to open it in the workspace. Similarly, if you click an object in a diagram, the Call Stack window highlights the object.

2.17.4.3.2 Trace window

The Trace window displays the debugger's output status and errors. For example:

```
11:22:06 AM> Job <GO> has been launched in debug mode.
11:22:07 AM> Breakpoint fired. Flow <aaa>: from <Query> to <"NewFile_773"(Simple)>
```

When the job completes, this window displays the following:

Job **`<JobName>`** finished. Stop debugger.

When the job completes the debugger gives you a final opportunity to examine data. When you must exit the debugger, select the *Stop Debug* button on the tool bar, press *shift*+*F8*, or select *DebugStop Debug*.

2.17.4.3.3 Debug Variables window

The Debug Variables window displays global variables in use by the job at each breakpoint.

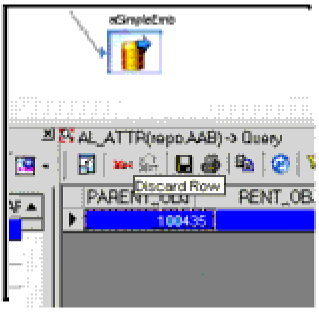
2.17.4.3.4 View Data pane

The View Data pane for lines uses the same tool bar and navigation options described for the View Data feature.

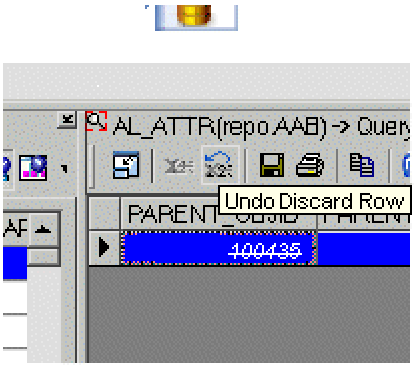
The following View Data pane options are unique to the interactive debugger:

- Allows you to view data that passes through lines.
- Displays (above the View Data tool bar) the names of objects to which a line connects using the format: **`<TableName>`**(**`<DatastoreName>`**.**`<TableOwnerName>`**)-> **`<QueryName>`**.
- Displays data one row at a time by default.
- Provides the *All* check box which allows you to see more than one row of processed data.

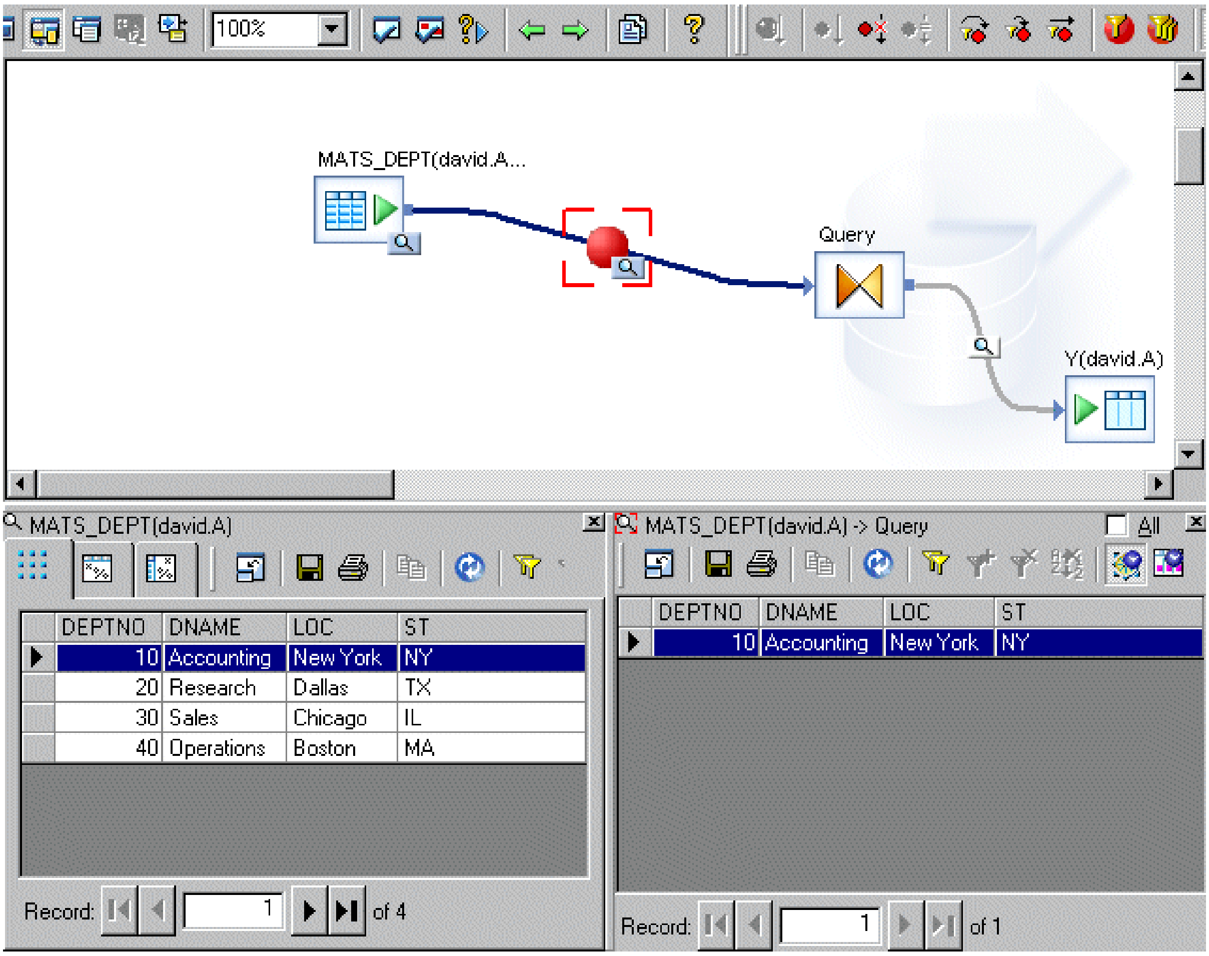
- Allows you to edit data in a cell.
  For example, You might want to fix an error temporarily to continue with a debugger run. You can fix the job design later to eliminate the error permanently.
  To edit cell data:
  - Deselect the *All* check box so that only has one row displayed.
  - Double-click a cell or right-click it and select *Edit cell*.
- Uses a property called the *Data sample rate*.
- Allows you to flag a row that you do not want the next transform to process.
  To discard a row from the next step in a data flow process, select it and click *Discard Row*.



Discarded row data appears in the strike-through style in the View Data pane (for example, 100345).
If you accidentally discard a row, you can undo the discard immediately afterwards. Select the discarded row and click *Undo Discard Row*.



Alternatively, right-click a row and select either Discard Row or *Undo Discard Row* from the shortcut menu.
For example, if a source in a data flow has four rows and you set the Data sample rate to 2 when you start the debugger, it displays the first row processed at a pre-defined breakpoint.

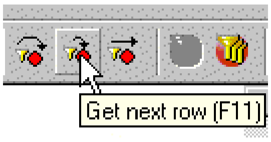


DEPTNO	DNAME	LOC	ST
10	Accounting	New York	NY
20	Research	Dallas	TX
30	Sales	Chicago	IL
40	Operations	Boston	MA

Record: 1 of 4

DEPTNO	DNAME	LOC	ST
10	Accounting	New York	NY

Record: 1 of 1



If you use the *Get Next Row* option, then the next row at the same breakpoint is displayed.

If you want to see both rows, select the *All* check box on the upper-right corner of this pane. The row displayed at the bottom of the table is the last row processed.

At this point, you have viewed two rows that have passed through a line.

If you click *Get Next Row* again, only the last two rows processed are displayed because you set the sample size to 2.

.

Related Information

Using View Data [page 688]

2.17.4.3.5 Filters and Breakpoints window

You can manage interactive debugger filters and breakpoints using the Filters/Breakpoints window. You can open this window from the Debug menu or tool bar.

Lines that contain filters or breakpoints are listed in the far-left side of the Filters/Breakpoints window. To manage these, select the line(s) that you want to edit, select a command from the list and click Execute. You can also select a single line on the left and view/edit its filters and breakpoints on the right side of this window. When you are finished using the Filters/Breakpoints window, click OK.

2.17.4.4 Debug menu options and tool bar

Once you start the interactive debugger, you can access appropriate options from the Designer's *Debug* menu and tool bar.

Image	Option	Description	Key Commands
	Execute	Opens the Execution Properties window from which you can select job properties then execute a job outside the debug mode. Available when a job is active in the workspace.	F8
	Start debug	Opens the Debug Properties window from which you can select job properties then execute a job in debug mode (start the debugger). Other Designer operations are set to read-only until you stop the debugger. Available when a job is active in the workspace.	Ctrl+F8
	Stop debug	Stops a debug mode execution and exits the debugger. All Designer operations are reset to read/write.	Shift+F8
	Pause debug	Allows you to manually pause the debugger. You can use this option instead of a breakpoint.	None
	Step over	Allows you to manually move to the next line in a data flow by stepping over a transform in the workspace. Use this option to see the first row in a data set after it is transformed. The workspace displays a red square on the line to indicate the path you are using. If the transform you step over has multiple outputs, the Designer provides a popup menu from which you can select the logic branch you want to take.	F10
	Get next row	Allows you to stay at the current breakpoint and view the next row of data in the data set.	F11
	Continue	Allows you to give control of the job back to the Designer. The debugger continues until: • You use the Pause debug option. • Another breakpoint is encountered.	Ctrl+F10

Image	Option	Description	Key Commands
		• The job completes.	
	Show Filters/ Breakpoints	Shows all filters and breakpoints that exist in a job. When not selected, all filters and breakpoints are hidden from view. This option is always available in the Designer.	None
	Set Filter/ Breakpoints...	Opens a dialog from which you can set, remove, edit, enable or disable filters and breakpoints. You can also set conditions for breakpoints. From the workspace, you can right-click a line and select the same option from a short cut menu. Available when a data flow is active in the workspace.	F9
	Filters/Breakpoints...	Opens a dialog with which you can manage multiple filters and breakpoints in a data flow. Also offers the same functionality as the Set Filters/Breakpoints window. This option is always available in the Designer.	Alt+F9
	Call Stack	Shows or hides the Call Stack window.	None
	Variables	Shows or hides the Debug Variables window.	None
	Trace	Shows or hides the Trace window.	None

2.17.4.5 Viewing data passed by transforms

To view the data passed by transforms, execute the job in debug mode.

2.17.4.5.1 To view data passed by transforms

1. In the project area, right-click a job and click *Start debug*.

   The Debug Properties window opens.
2. Clear the *Exit the debugger when the job is finished* check box.
3. You can enter a value in the *Data sample rate* text box or leave the default value, which is 500.
4. Click *OK*.

2.17.4.5.2 To view sample data in debug mode

1. While still in debug mode after the job completes, in the project area, click the name of the data flow to view.
2. Click the View Data button displayed on a line in the data flow.
3. Navigate through the data to review it. When done, click the *Stop debug* button on the toolbar.

2.17.4.6 Push-down optimizer

When the software executes a job, it normally pushes down as many operations as possible to the source database to maximize performance. Because the interactive debugger requires a job execution, the following push-down rules apply:

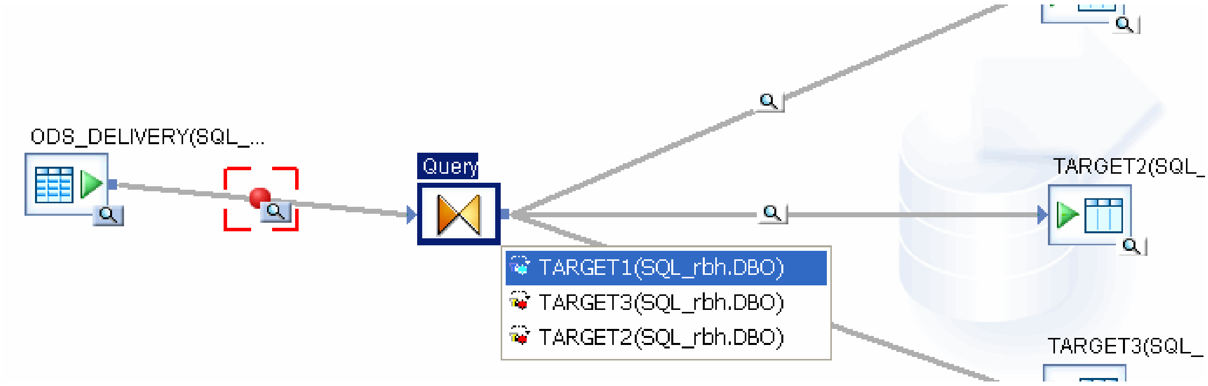
- Query transforms
  The first transform after a source object in a data flow is optimized in the interactive debugger and pushed down to the source database if both objects meet the push-down criteria and if you have not placed a breakpoint on the line before the first transform.
- Breakpoints
  The software does not push down any operations if you set a pre-defined breakpoint. Pre-defined breakpoints are breakpoints defined before you start the interactive debugger.
  After the interactive debugger is started, if the first transform is pushed down, the line is disabled during the debugging session. You cannot place a breakpoint on this line and you cannot use the View Data pane.
- Filters
  If the input of a pre-defined filter is a database source, it is pushed down. Pre-defined filters are interactive debugger filters defined before you start the interactive debugger.

Related Information

Performance Optimization Guide: Push-down operations [page 2116]

2.17.4.7 Limitations

- The interactive debugger can be used to examine data flows. Debug options are not available at the work flow level.
- A repository upgrade is required to use this feature.
- The debugger cannot be used with ABAP data flows.
- All objects in a data flow must have a unique name. For example, if there are several outputs for a transform you can choose which path to use. If any of these objects have the same name, the result of your selection is unpredictable.



2.17.5 Comparing Objects

The software allows you to compare any two objects and their properties by using the Difference Viewer utility. You can compare:

- two different objects
- different versions of the same object
- an object in the local object library with its counterpart in the central object library

You can compare just the top-level objects, or you can include the object's dependents in the comparison.

Objects must be of the same type; for example, you can compare a job to another job or a custom function to another custom function, but you cannot compare a job to a data flow.

2.17.5.1 To compare two different objects

1. In the local or central object library, right-click an object name.
2. From the shortcut menu, highlight *Compare*, and from the submenu, click one of the following options (availability depends on the object you selected):

Option	Description
Object to central	Compares the selected object to its counterpart in the central object library.
Object with dependents to central	Compares the selected object and its dependent objects to its counterpart in the central object library.
Object to...	Compares the selected object to another similar type of object.
Object with dependents to...	Compares the selected object and its dependents to another similar type of object.

   The cursor changes to a target icon.
3. When you move the cursor over an object that is eligible for comparison, the target cursor changes color. Click on the desired object.

The *Difference Viewer* window opens in the workspace.

The window identifies changed items with a combination of icons, color, and background shading. Some of these properties are configurable.

Depending on the object type, the panes show items such as the object's properties and the properties of and connections (links) between its child objects.

2.17.5.2 To compare two versions of the same object

If you are working in a multiuser environment and using a central object library, you can compare two objects that have different versions or labels.

1. In the central object library, right-click an object name, and from the shortcut menu click *Show History*.
2. In the *History* window, Ctrl-click the two versions or labels you want to compare.
3. Click *Show Differences* or *Show Differences with Dependents*.

   The *Difference Viewe*r window opens in the workspace.
4. Close the *History* window.

Related Information

Designer Guide: Working in a Multi-user Environment, Viewing object history [page 822]

2.17.5.3 Overview of the Difference Viewer window

The first object you selected appears in the left pane of the window, and the second object appears on the right. Following each object name is its location.

The Difference Viewer window includes the following features:

- toolbar
- navigation bar
- status bar
- shortcut menu

Also, when a Difference Viewer window is active, the main designer window also contains a menu called Difference Viewer. The next section describes these features.

You can have multiple Difference Viewer windows open at a time in the workspace. To refresh a Difference Viewer window, press F5.

Expanding or collapsing any property set also expands or collapses the compared object's corresponding property set.

2.17.5.3.1 Toolbar

The toolbar includes the following buttons.

Navigation buttons

- First Difference (Alt+Home)
- Previous Difference (Alt+left arrow)
- Current Difference
- Next Difference (Alt+right arrow)
- Last Difference (Alt+End)

Filter buttons

- Enable filter(s)—Click to open the Filters dialog box.
    - *Hide non-executable elements*—Select this option to remove from view those elements that do not affect job execution.
    - *Hide identical elements*—Select this option to remove from view those elements that do not have differences.
- Disable filters—Removes all filters applied to the comparison.

Show levels

Show Level 1 shows only the objects you selected for comparison, Show Level 2 expands to the next level, etc. Show All Levels expands all levels of both trees.

Find (Ctrl+F)

Click *Find* to open a text search dialog box.

Open in new window

Click to open the currently active Difference Viewer in a separate window. You must close this window before continuing.

2.17.5.3.2 Navigation bar

The vertical navigation bar contains colored bars that represent each of the differences throughout the comparison. The colors correspond to those in the status bar for each difference. An arrow in the navigation bar indicates the difference that is currently highlighted in the panes. You can click on the navigation bar to select a difference (the cursor point will have a star on it). The purple brackets in the bar indicate the portion of the comparison that is currently in view in the panes.

See the next section for more information on how to navigate through differences.

2.17.5.3.3 Status bar

The status bar at the bottom of the window includes a key that illustrates the color scheme and icons that identify the differences between the two objects.

Icon	Difference	Description
✖	Deleted	The item does not appear in the object in the right pane.
▲	Changed	The differences between the items are highlighted in blue (the default) text.
➡	Inserted	The item has been added to the object in the right pane.
!	Consolidated	This icon appears next to an item if items within it have differences. Expand the item by clicking its plus sign to view the differences

You can change the color of these icons by right-clicking in the Difference Viewer window and clicking Configuration.

The status bar also includes a reference for which difference is currently selected in the comparison (for example, the currently highlighted difference is 9 of 24 total differences in the comparison).

The status bar also indicates that there is at least one filter applied to this comparison.

Related Information

Shortcut menu [page 717]

2.17.5.3.4 Shortcut menu

Right-clicking in the conbody of the Difference Viewer window displays a shortcut menu that contains all the toolbar commands plus:

- View — Toggle to display or hide the status bar, navigation bar, or secondary toolbar (an additional toolbar that appears at the top of the window; you might find this useful if you have the Differences Viewer open in a separate window).
- Layout — Use to reposition the navigation bar.
- Configuration — Click to modify viewing options for elements with differences.

Related Information

To change the color scheme [page 718]

2.17.5.3.4.1 To change the color scheme

The status bar at the bottom of the Difference Viewer window shows the current color scheme being used to identify deleted, changed, inserted, or consolidated items in the comparison panes. You can customize this color scheme as follows.

1. Right-click in the conbody of the Difference Viewer window to display the shortcut toolbar.
2. Click Configuration to open the Configuration window.
3. Click a marker (Inserted, Deleted, Changed, or Consolidated) to change.
4. Click the Color sample to open the Color palette.
5. Click a Basic color or create a custom color.
6. Click *OK*.
7. Click another marker to change it, or click *OK* to close the Configuration window.

2.17.5.3.4.2 To change the background shading

Items with differences appear with a background default color of grey. You can customize this background.

1. Right-click in the conbody of the Difference Viewer window to display the shortcut toolbar.
2. Click Configuration to open the Configuration window.
3. Click a marker to change, or select the *Apply for all markers* check box.
4. Click the Background sample to open the Color palette.
5. Click a Basic color or create a custom color.
6. Click *OK*.
7. To apply different background colors to different markers, click the marker to configure and repeat steps 4 through 6.
8. Click *OK* to close the Configuration window.

2.17.5.3.5 Difference Viewer menu

When a Difference Viewer window is active in the workspace, the main Designer window contains a menu called Difference Viewer. The menu contains the same commands as the toolbar.

2.17.5.4 Navigating through differences

The Difference Viewer window offers several options for navigating through differences.

You can navigate through the differences between the objects using the navigation buttons on the toolbar. For example, clicking the Next Difference button highlights the next item that differs in some way from the compared object. The item is marked with the appropriate icon and only the differing text appears highlighted in the color assigned to that type of difference.

You can also use the navigation bar. Select an item in either pane that has a difference. Note that an arrow appears next to the colored bar that corresponds to that item. You can click on these bars to jump to different places in the comparison, for example to view only inserted items (with a default color of green). The purple brackets in the bar indicate the portion of the comparison that is currently in view in the panes. Use the scroll bar in either pane to adjust the bracketed view.

For text-based items such as scripts, click the magnifying glass to view the text in a set of new panes that appear below the main object panes. Use the scroll bars for these panes to navigate within them. Click the magnifying glass (or any other item) to close the text panes.

2.17.6 Calculating column mappings

SAP Data Services can calculate information about target tables and columns and the sources used to populate them, for example for impact and lineage or auto documentation reports.

Calculating column mappings populates the internal ALVW_MAPPING view and the AL_COLMAP_NAMES table. The ALVW_MAPPING view provides current data to metadata reporting applications like Impact and Lineage Analysis. If you need to generate a report about a data flow that processes nested (NRDM) data, query the AL_COLMAP_NAMES table using a custom report.

Whenever a column mapping calculation is in progress, the Designer displays a status icon at the bottom right of the window. You can double-click this icon to cancel the process.

To calculate column mappings, you can:

- Enable the option in the Designer to automatically calculate column mappings.
- Execute the column mapping process manually from either the Designer or the Impact and Lineage Analysis application in the Management Console.

Related Information

Reference Guide: Metadata in Repository Tables and Views, Storing nested column-mapping data [page 1731]

2.17.6.1 To automatically calculate column mappings

To set the option to automatically calculate column mapping information, in the Designer select ▶ *Tools* > *Options* > *Designer* > *General* > *Automatically calculate column mappings* ◀. This option is selected by default.

Note that if the Designer option *Automatically calculate column mappings* is cleared, any subsequent changes made to the data flow require that you manually recalculate the column mappings to ensure the ALVW_MAPPING view and the AL_COLMAP_NAMES table have the most current information.

2.17.6.2 To manually calculate column mappings

If the Designer option *Automatically calculate column mappings* is cleared and you want to generate reports, you can manually calculate the mappings. You can manually calculate column mappings at any time in either the Designer or the Management Console.

In the Designer, right-click in the object library and select ▶ *Repository* > *Calculate column mappings* ◀.

In the Management Console:

1. Select *Impact and Lineage Analysis*.
2. Open the *Settings* control panel.
3. Click the *Refresh Usage Data* tab.
4. Select the Job Server that is associated with the repository you want to use.
5. Click *Calculate Column Mapping*.

On the Impact and Lineage Analysis *Overview* tab, you can expand *Data Flow Column Mapping Calculation* to view a list of data flows and the calculation status of each. If the mapping calculation is complete, the *Status* indicator is checked.

2.18 Recovery Mechanisms

Recovery mechanisms are available in SAP Data Services for batch jobs only.

Related Information

Recovering from unsuccessful job execution [page 721]
Automatically recovering jobs [page 721]
Manually recovering jobs using status tables [page 728]
Processing data with problems [page 729]

2.18.1 Recovering from unsuccessful job execution

If an SAP Data Services job does not complete properly, you must fix the problems that prevented the successful execution of the job and run the job again.

However, during the failed job execution, some data flows in the job may have completed and some tables may have been loaded, partially loaded, or altered. Therefore, you need to design your data movement jobs so that you can recover—that is, rerun the job and retrieve all the data without duplicate or missing data.

You can use various techniques to recover from unsuccessful job executions. This section discusses two techniques:

- Automatically recovering jobs — A software feature that allows you to run unsuccessful jobs in recovery mode.
- Manually recovering jobs using status tables — A design technique that allows you to rerun jobs without regard to partial results in a previous run.

You might need to use a combination of these techniques depending on the relationships between data flows in your application.

If you do not use these techniques, you might need to roll back changes manually from target tables if interruptions occur during job execution.

Related Information

Automatically recovering jobs [page 721]

Manually recovering jobs using status tables [page 728]

2.18.2 Automatically recovering jobs

With automatic recovery, the software records the result of each successfully completed step in a job. If a job fails, you can choose to run the job again in recovery mode. During recovery mode, the software retrieves the results for successfully completed steps and reruns uncompleted or failed steps under the same conditions as the original job. For recovery purposes, the software considers steps that raise exceptions as failed steps, even if the step is caught in a try/catch block.

2.18.2.1 Enabling automated recovery

To use the automatic recover feature, you must enable the feature during initial execution of a job. The software saves the results from successfully completed steps when the automatic recovery feature is enabled.

2.18.2.1.1 To run a job from Designer with recovery enabled

1. In the project area, select the job name.
2. Right-click and choose *Execute*.

   The software prompts you to save any changes.
3. Make sure that the *Enable Recovery* option is selected in the *Execution Properties* window.

   If this check box is not selected, the software does not record the results from the steps during the job and cannot recover the job if it fails. In that case, you must perform any recovery operations manually.

2.18.2.1.2 To run a job with recovery enabled from the Administrator

When you schedule or execute a job from the Administrator, select the *Enable Recovery* check box.

2.18.2.2 Marking recovery units

In some cases, steps in a work flow depend on each other and must be executed together. Because of the dependency, you should designate the work flow as a "recovery unit." When a work flow is a recovery unit, the entire work flow must complete successfully. If the work flow does not complete successfully, the software executes the entire work flow during recovery, even steps that executed successfully in prior work flow runs.

However, there are some exceptions to recovery unit processing. For example, when you specify that a work flow or a data flow should only execute once, a job will never re-execute that work flow or data flow after it completes successfully, except if that work flow or data flow is contained within a recovery unit work flow that re-executes and has not completed successfully elsewhere outside the recovery unit.

It is recommended that you not mark a work flow or data flow as *Execute only once* when the work flow or a parent work flow is a recovery unit.

Related Information

Reference Guide: Data flow [page 857]
Reference Guide: Work flow [page 1002]

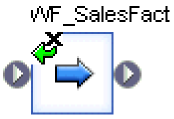
2.18.2.2.1 To specify a work flow as a recovery unit

1. In the project area, select the work flow.
2. Right-click and choose *Properties*.

3. Select the *Recover as a unit* check box, then click *OK*.

   During recovery, the software considers this work flow a unit. If the entire work flow completes successfully—that is, without an error—during a previous execution, then the software retrieves the results from the previous execution. If any step in the work flow did not complete successfully, then the entire work flow re-executes during recovery.

   On the workspace diagram, the black "x" and green arrow symbol indicate that a work flow is a recovery unit.



2.18.2.3 Running in recovery mode

If a job with automated recovery enabled fails during execution, you can re-execute the job in recovery mode.

As with any job execution failure, you need to determine and remove the cause of the failure and rerun the job in recovery mode. If you need to make any changes to the job itself to correct the failure, you cannot use automatic recovery but must run the job as if it is a first run.

In recovery mode, the software executes the steps or recovery units that did not complete successfully in a previous execution—this includes steps that failed and steps that threw an exception but completed successfully such as those in a try/catch block. As in normal job execution, the software executes the steps in parallel if they are not connected in the work flow diagrams and in serial if they are connected.

2.18.2.3.1 To run a job in recovery mode from Designer

1. In the project area, select the (failed) job name.
2. Right-click and choose *Execute*.

   The software prompts you to save any objects that have unsaved changes.
3. Make sure that the *Recover from last failed execution* check box is selected.

   This option is not available when a job has not yet been executed, when the previous run succeeded, or when recovery mode was disabled during previous run.

   When you select Recover from last failed execution, the software retrieves the results from any steps that were previously executed successfully and executes or re-executes any other steps.

   If you clear this option, the software runs this job anew, performing all steps.

When you schedule or execute a (failed) job from the Administrator, select the *Recover from last failed execution* check box.

2.18.2.4 Ensuring proper execution path

The automated recovery system requires that a job in recovery mode runs again exactly as it ran previously.

It is important that the recovery job run exactly as the previous run. If the job was allowed to run under changed conditions—suppose a `<sysdate>` function returns a new date to control what data is extracted—then the new data loaded into the targets will no longer match data successfully loaded into the target during the first execution of the job.

For example, suppose a daily update job running overnight successfully loads dimension tables in a warehouse. However, while the job is running, the database log overflows and stops the job from loading fact tables. The next day, the administrator truncates the log file and runs the job again in recovery mode. The recovery job does not reload the dimension tables because the original, failed run successfully loaded them.

To ensure that the fact tables are loaded with the data that corresponds properly to the data already loaded in the dimension tables, the recovery job must use the same extraction criteria that the original job used when loading the dimension tables. If the recovery job used new extraction criteria—such as basing data extraction on the current system date—the data in the fact tables would not correspond to the data previously extracted into the dimension tables.

In addition, if the recovery job uses new values, then the job execution may follow a completely different path through conditional steps or try/catch blocks.

If any global variable value is changed within a recovery as unit work flow, any downstream global variable reference is not guaranteed to have the updated value in the recovery mode execution. This will happen in the recovery mode execution if the previous execution failure is not in the work flow that contains the variable change.

When recovery is enabled, the software stores results from the following types of steps:

- Work flows
- Batch data flows
- Script statements
- Custom functions (stateless type only)
- SQL function
- exec function
- get_env function
- rand function
- sysdate function
- systime function

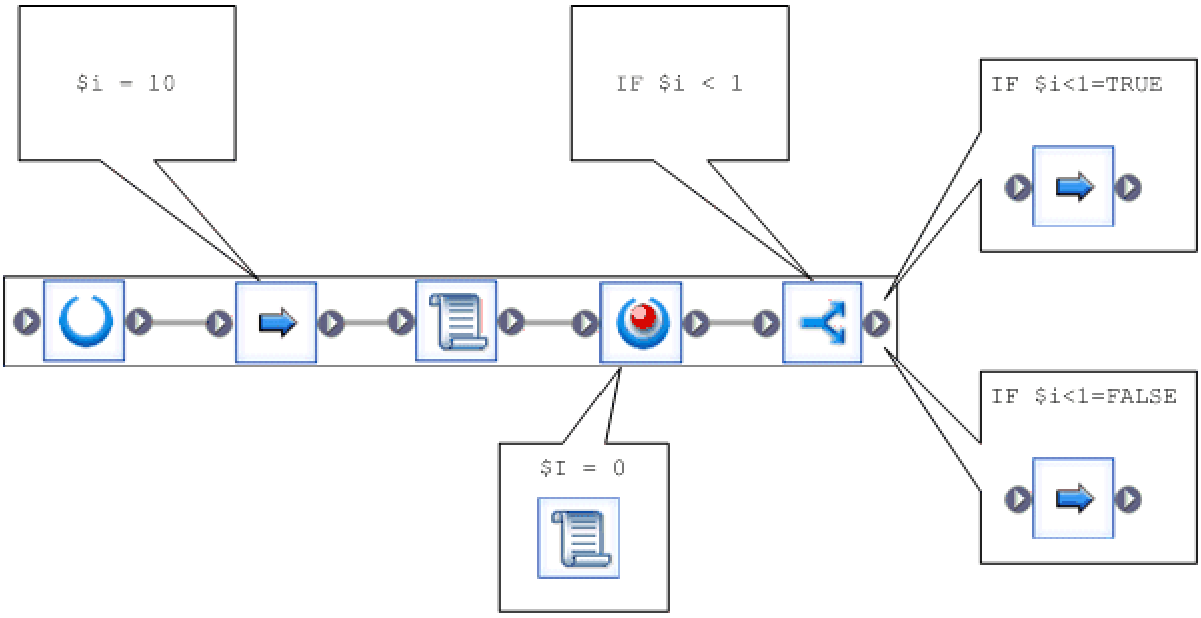
2.18.2.5 Using try/catch blocks with automatic recovery

SAP Data Services does not save the result of a try/catch block for reuse during recovery. If an exception is thrown inside a try/catch block, then during recovery the software executes the step that threw the exception and subsequent steps.

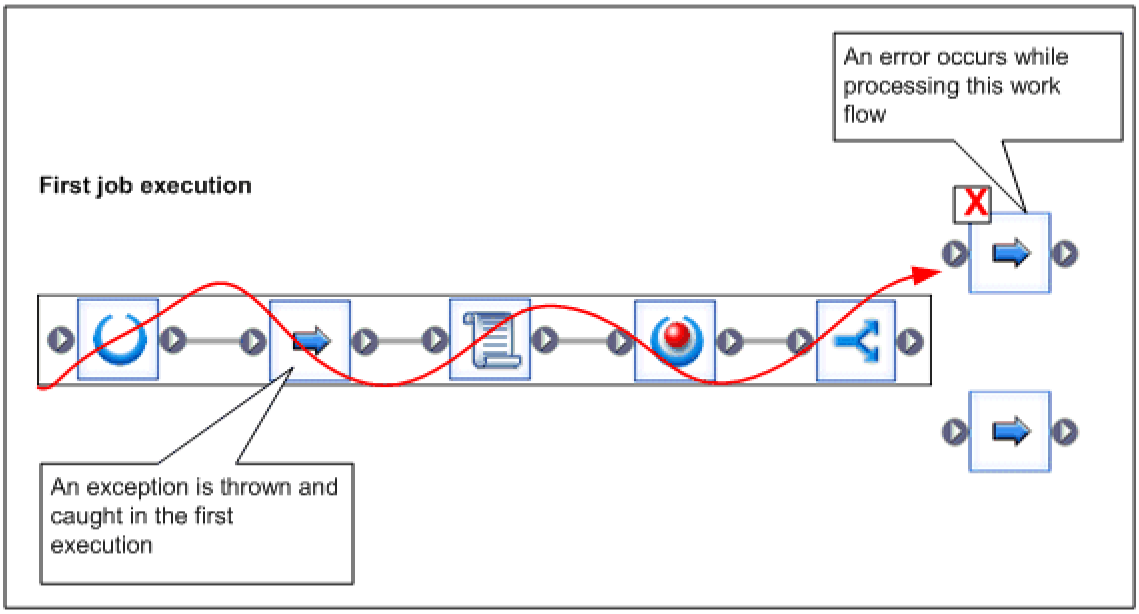
Because the execution path through the try/catch block might be different in the recovered job, using variables set in the try/catch block could alter the results during automatic recovery.

For example, suppose you create a job that defines a variable, `$i,` that you set within a try/catch block. If an exception occurs, you set an alternate value for `$i`. Subsequent steps are based on the value of `$i`.

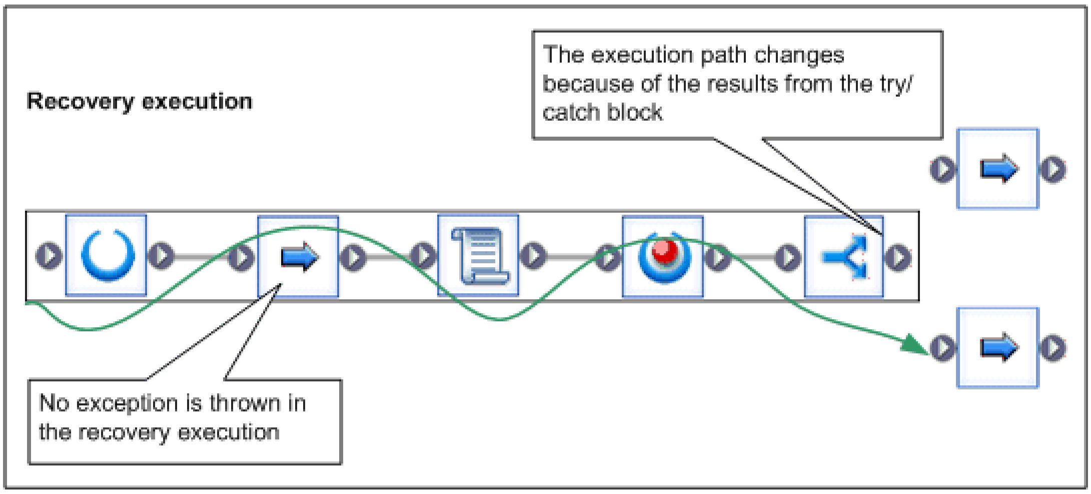
Job execution logic



During the first job execution, the first work flow contains an error that throws an exception, which is caught. However, the job fails in the subsequent work flow.


First job execution
An error occurs while processing this work flow
X
An exception is thrown and caught in the first execution

You fix the error and run the job in recovery mode. During the recovery execution, the first work flow no longer throws the exception. Thus the value of the variable, `<$i>` , is different, and the job selects a different subsequent work flow, producing different results.


Recovery execution
The execution path changes because of the results from the try/ catch block
No exception is thrown in the recovery execution

To ensure proper results with automatic recovery when a job contains a try/catch block, do not use values set inside the try/catch block in any subsequent steps.

2.18.2.6 Ensuring that data is not duplicated in targets

Define work flows to allow jobs correct recovery. A data flow might be partially completed during an incomplete run. As a result, only some of the required rows could be inserted in a table. You do not want to insert duplicate rows during recovery when the data flow re-executes.

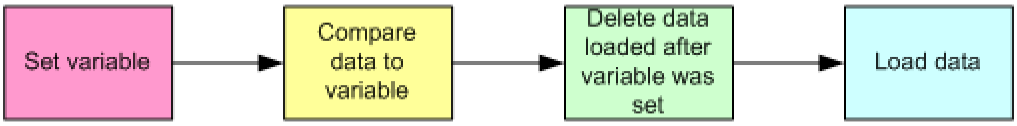
You can use several methods to ensure that you do not insert duplicate rows:

- Design the data flow to completely replace the target table during each execution
  This technique can be optimal when the changes to the target table are numerous compared to the size of the table. You can use tuning techniques such as bulk loading options to improve overall performance.
- Set the auto correct load option for the target table
  The auto correct load option checks the target table for existing rows before adding new rows to the table. Using the auto correct load option, however, can needlessly slow jobs executed in non-recovery mode. Consider this technique when the target table is large and the changes to the table are relatively few.
- Include a SQL command to execute before the table loads
  Preload SQL commands can remove partial database updates that occur during incomplete execution of a step in a job. Typically, the preload SQL command deletes rows based on a variable that is set before the partial insertion step began.

2.18.2.7 Using preload SQL to allow re-executable data flows

To use preload SQL commands to remove partial database updates, tables must contain a field that allows you to tell when a row was inserted. Create a preload SQL command that deletes rows based on the value in that field.

For example, suppose a table contains a column that records the time stamp of any row insertion. You can create a script with a variable that records the current time stamp before any new rows are inserted. In the target table options, add a preload SQL command that deletes any rows with a time-date stamp greater than that recorded by the variable.



During initial execution, no rows match the deletion criteria. During recovery, the variable value is not reset. (The variable value is set in a script, which is executed successfully during the initial run.) The rows inserted during the previous, partial database load would match this criteria, and the preload SQL command would delete them.

To use preload SQL commands properly, you must define variables and pass them to data flows correctly.

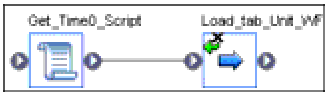
2.18.2.7.1 To use preload SQL commands to ensure proper recovery

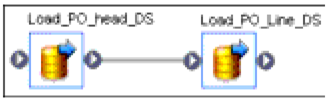
1. Determine appropriate values that you can use to track records inserted in your tables.

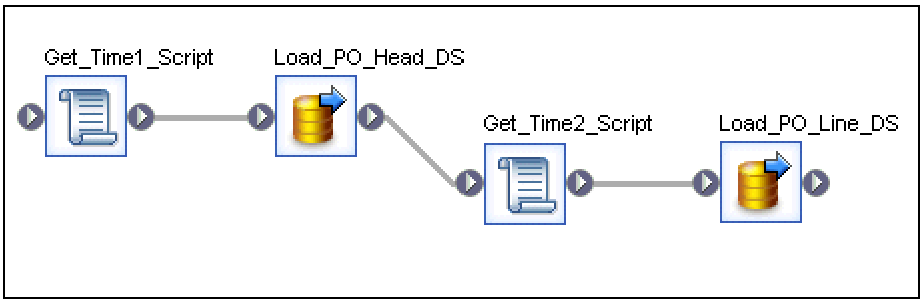
   For example, if each row in a table is marked with the insertion time stamp, then you can use the value from the `sysdate()` function to determine when a row was added to that table.
2. Create variables that can store the "tracking" values.

   Variables are either job or work-flow specific. If a work flow is a recovery unit, create the "tracking" variables for that work flow at the job level; otherwise, create your tracking variables at the work flow level. Generally, you do not want tracking variables reset during recovery because when they reset, the preload SQL command will not work properly.
3. Create scripts that set the variables to the appropriate values.

   Scripts are unique steps in jobs or work flows. You need to create a separate script that sets the required variables before each data flow or work flow that loads a table. If a work flow is a recovery unit, create the scripts for that work flow at the job level; otherwise, create the scripts at the work flow level.
4. Connect the scripts to the corresponding data flows or work flows.







5. Create parameters to pass the variable information from the job or work flow where you created the variable to the data flow that uses the tracking variable in the preload SQL command.
6. Insert appropriate preload SQL commands that remove any records inserted during earlier unsuccessful runs.

   The preload SQL commands reference the parameter containing the tracking variable, deleting rows that were inserted after the variable was set.

   For example, suppose the **`<PO_ITEM>`** table records the date-time stamp in the **`<TIMESTMP>`** column. You created a variable **`<$load_time>`** that records the value from the `sysdate()` function before the load starts, and you passed that variable to the data flow that loads the **`<PO_ITEM>`** table in a parameter named **`<$load_time>`**. Then, your preload SQL command must delete any records in the table where the value in **`<TIMESTMP>`** is larger than the value in **`<$load_time>`**.

   Delete from **`<PO_ITEM>`** where **`<TIMESTMP>`** `> [$`**`<load_time>`**`]`

Related Information

To define a local variable [page 404]
Scripts [page 341]
Defining parameters [page 405]

2.18.3 Manually recovering jobs using status tables

You can design your jobs and work flows so that you can manually recover from an unsuccessful run. A job designed for manual recovery must have certain characteristics:

- You can run the job repeatedly.
- The job implements special steps to recover data when a step did not complete successfully during a previous run.

You can use an execution status table to produce jobs that can be run multiple times without duplicating target rows. The table records a job's execution status. A "failure" value signals SAP Data Services to take a recovery execution path.

To implement a work flow with a recovery execution path:

- Define a flag that indicates when the work flow is running in recovery mode.

- Store the flag value in a status table.
- Check the flag value in the status table before executing a work flow to determine which path to execute in the work flow.
- Update the flag value when the work flow executes successfully.

For example, you could design a work flow that uses the auto correct load option when a previous run does not complete successfully. This work flow would have five steps, as illustrated:

1. Retrieve the flag value, which indicates the success or failure of the previous execution, from the status table. Store this value in a variable such as `<$recovery_needed>` .
2. In a conditional, evaluate the `<$recovery_needed>` variable.
3. If recovery is required, execute the recovery data flow recover_customer. This data flow loads the data using the auto correct load option.
4. If recovery is not required, execute the non-recovery data flow load_customer. This data flow loads the data without the auto correct load option.
5. Update the flag value in the status table to indicate successful execution.

Related Information

Reference Guide: Target [page 960]

2.18.4 Processing data with problems

Jobs might not produce the results you expect because of problems with data. In some cases, the software is unable to insert a row. In other cases, the software might insert rows with missing information. You can design your data flows to anticipate and process these types of problems. For example, you might have a data flow write rows with missing information to a special file that you can inspect later.

This section describes mechanisms you can use to anticipate and process data problems. In particular, this section discusses three techniques:

- Using overflow files
- Filtering missing or bad values
- Handling facts with missing dimensions

2.18.4.1 Using overflow files

A row that cannot be inserted is a common data problem. Use the overflow file to process this type of data problem. When you specify an overflow file and the SAP Data Services cannot load a row into a table, the software writes the row to the overflow file instead. The trace log indicates the data flow in which the load failed and the location of the file.

For any table used as a target, you can set the option to use an overflow file in the *Options* tab. When you specify an overflow file, give a full path name to ensure that the software creates a unique file when more than one file is created in the same job. By default, the name of the overflow file is the target table name.

When you select the overflow file option, you choose what the software writes to the file about the rows that failed to load: either the data from the row or the SQL commands required to load the row. If you select data, you can use the software to read the data from the overflow file, cleanse it, and load it into the target table. If you select SQL commands, you can use the commands to load the target manually when the target is accessible.

There are many reasons for loading to fail, for example:

- Out of memory for the target
- Overflow column settings
- Duplicate key values

You can use the overflow information to identify invalid data in your source or problems introduced in the data movement. Every new run will overwrite the existing overflow file.

> i Note
>
> You cannot use overflow files when loading to a BW Transfer Structure.

2.18.4.2 Filtering missing or bad values

A missing or invalid value in the source data is another common data problem. Using queries in data flows, you can identify missing or invalid values in source data. You can also choose to include this data in the target or to disregard it.

For example, suppose you are extracting data from a source and you know that some phone numbers and customer names are missing. You can use a data flow to extract data from the source, load the data into a target, and filter the NULL values into a file for your inspection.

The data flow has five steps:

1. Extracts data from the source.
2. Selects the data set to load into the target and applies new keys. (It does this by using the Key_Generation function.)
3. Loads the data set into the target, using the bulk load option for best performance.
4. Uses the same data set for which new keys were generated in step 2, and select rows with missing customer names and phone numbers.
5. Writes the customer IDs for the rows with missing data to a file.

Now, suppose you do not want to load rows with missing customer names into your target. You can insert another query into the data flow to ensure that SAP Data Services does not insert incomplete rows into the target. The new query filters the rows with missing customer names before loading any rows into the target. The missing data query still collects those rows along with the rows containing missing phone numbers. In this version of the example, the Key_Generation transform adds keys for new rows before inserting the filtered data set into the target.

The data flow now has six steps.

1. Extracts data from the source.
2. Selects the data set to load into the target by filtering out rows with no customer name values.
3. Generates keys for rows with customer names.
4. Loads the valid data set (rows with customer names) into the target using the bulk load option for best performance.

5. Uses a separate query transform to select rows from the source that have no names or phones.
   Note that the software does not load rows with missing customer names into the target; however, the software does load rows with missing phone numbers.
6. Writes the customer IDs for the rows with missing data to a file.

You could add more queries into the data flow to select additional missing or invalid values for later inspection.

2.18.4.3 Handling facts with missing dimensions

Another data problem occurs when SAP Data Services searches a dimension table and cannot find the values required to complete a fact table.

You can approach this problem in several ways:

- Leave the problem row out of the fact table.
  Typically, this is not a good idea because analysis done on the facts will be missing the contribution from this row.
- Note the row that generated the error, but load the row into the target table anyway.
  You can mark the row as having an error, or pass the row information to an error file as in the examples from *Filtering missing or bad values* [page 730].
- Fix the problem programmatically.
  Depending on the data missing, you can insert a new row in the dimension table, add information from a secondary source, or use some other method of providing data outside of the normal, high-performance path.

2.18.5 Exchanging metadata

SAP Data Services offers several methods for exchanging metadata:

- Using the Metadata Exchange option, you can export metadata into an XML file. After you create the file, you must manually import it into another tool.
- Using the SAP BusinessObjects Universes option, you can export metadata directly from a repository into a universe using the *Create* or *Update* data mode.

Related Information

Metadata exchange [page 731]

Creating BusinessObjects universes [page 733]

2.18.5.1 Metadata exchange

You can exchange metadata between the software and third-party tools using XML files and the *Metadata Exchange* option.

The software supports two built-in metadata exchange formats:

- CWM 1.0 XML/XMI 1.1
  CWM (the Common Warehouse Metamodel)— is a specification that enables easy interchange of data warehouse metadata between tools, platforms, and repositories in distributed heterogeneous environments.
- ERwin 4.x XML

The software can also use:

- MIMB (the *Meta Integration® Model Bridge*)
  MIMB is a Windows stand-alone utility that converts metadata models among design tool formats. By using MIMB with the software, you can exchange metadata with all formats that MIMB supports. If MIMB is installed, the additional formats it supports are listed in the Metadata Exchange window.
- BusinessObjects Universe Builder
  Converts repository metadata to BusinessObjects universe metadata. See *Creating BusinessObjects universes* [page 733].

2.18.5.1.1 Importing metadata files into the software

You can import metadata from ERwin Data Modeler 4.x XML into a datastore.

2.18.5.1.1.1 To import metadata using Metadata Exchange

1. From the *Tools* menu, select *Metadata Exchange*.
2. In the Metadata Exchange window, select *Import metadata from file*.
3. In the *Metadata format* box, select `ERwin 4.x XML` from the list of available formats.
4. Specify the *Source file name* (enter directly or click *Browse* to search).
5. Select the *Target datastore name* from the list of datastores.
6. Click *OK* to complete the import.

2.18.5.1.2 Exporting metadata files from the software

You can export metadata into a file that other tools can read.

2.18.5.1.2.1 To export metadata using Metadata Exchange

1. From the *Tools* menu, select *Metadata Exchange*.
2. In the Metadata Exchange window, select *Export Data Services metadata to file*.
3. Select a *Metadata format* for the target from the list of available formats.

If you have MIMB installed and you select an MIMB-supported format, select the Visual check box to open the MIMB application when completing the export process.

If you do not select the Visual check box, the metadata is exported without opening the MIMB application. Using the MIMB application provides more configuration options for structuring the metadata in the exported file.

4. Specify the target file name (enter directly or click *Browse* to search).

   When you search for the file, you open a typical browse window, like this:

   Find any of the following file formats/types:

Format	File type
DI CWM 1.0 XML/XMI 1.1	XML
DI ERwin 4.x XML	XML
MIMB format (only if installed)	All

   After you select a file, click Open.
5. Select the *Source datastore name* from the list of datastores.
6. Click *OK* to complete the export.

2.18.5.2 Creating BusinessObjects universes

The software allows you to easily export its metadata to BusinessObjects universes for use with business intelligence tools. A universe is a layer of metadata used to translate physical metadata into logical metadata. For example the physical column name `deptno` might become `Department Number` according to a given universe design.

i Note

To use this export option, first install BusinessObjects Universe Builder on the same computer as SAP BusinessObjects Designer and SAP Data Services Designer. You can install Universe Builder using the installer for Designer or using the separate Universe Builder CD.

You can create usinessObjects universes using the *Tools* menu or the object library.

2.18.5.2.1 To create universes using the Tools menu

1. Select *Tools* > *BusinessObjects Universes* .
2. Select either *Create* or *Update*.

   The *Create Universe* or *Update Universe* window opens.
3. Select the datastore(s) that contain the tables and columns to export and click *OK*.

   The software launches the Universe Builder application and provides repository information for the selected datastores.

For more information, refer to the BusinessObjects Universe Builder Guide.

2.18.5.2.2 To create universes using the object library

1. Select the *Datastores* tab.
2. Right-click a datastore and select *BusinessObjects Universes*.
3. Select either *Create* or *Update*.

   The software launches the Universe Builder application and provides repository information for the selected datastores.

   For more information, refer to the BusinessObjects Universe Builder Guide.

2.18.5.2.3 Mappings between repository and universe metadata

SAP Data Services metadata maps to BusinessObjects Universe metadata as follows:

SAP Data Services	BusinessObjects Universe
Table	Class, table
Column	Object, column
Owner	Schema
Column data type (see next table)	Object data type
Primary key/foreign key relationship	Join expression
Table description	Class description
Table Business Description	Class description
Table Business Name	Class name
Column description	Object description
Column Business description	Object description
Column Business Name	Object name
Column mapping	Object description
Column source information (lineage)	Object description

Data types also map:

Data type	BusinessObjects Type
Date/Datetime/Time	Date
Decimal	Number
Int	Number
Double/Real	Number
Interval	Number
Varchar	Character
Long	Long Text

2.18.5.2.4 Attributes that support metadata exchange

The attributes *Business_Name* and *Business_Description* exist in the software for both tables and columns. These attributes support metadata exchanged between SAP Data Services and SAP BusinessObjects Universe Builder.

- A Business_Name is a logical field. Data Services stores it as a separate and distinct field from physical table or column names. Use this attribute to define and run jobs that extract, transform, and load physical data while the Business Name data remains intact.
- A Business_Description is a business-level description of a table or column. Data Services transfers this information separately and adds it to a BusinessObjects Class description.

The software includes two additional column attributes that support metadata exchanged between SAP Data Services and SAP BusinessObjects Universe Builder:

- Column_Usage
- Associated_Dimension

Related Information

Reference Guide: Object options, properties, and attributes [page 834]

2.18.6 Loading Big Data file with recovery option

Loading big data file with recovery option is an extension of the existing recovery mechanism in Data Services. In earlier versions of Data Services, the recovery mechanism was only supported until the data flow layer. As a result, if the engine crashed at any data flow, the whole data flow had to be restarted from the beginning in recovery mode.

However, if there is big data inside of the failed dataflow, restarting the process will involve duplicating work of cleaning and restarting. This restarting situation becomes worse if there is a reason that the job needs to be stopped periodically (for example, due to a network disconnect, database timeout or machine reboot).

To avoid this situation, when reading a big source flat file into the database tables, you have the option to turn on the big data loading recovery feature. For example, when you load a large input file into a database, the job may take several days to finish. If the engine crashes or fails in the middle of a job, the job can be resumed from the last failed check point. The last failed check point includes the failed filename and the offset of the file that has been processed so far. The recovery feature avoids restarting of the data flow and ensures that loading progresses forward from where it failed.

2.18.6.1 Turning on the recovery option for Big Data loading

To use the automatic recovery feature for big data loading, you must enable the feature during the initial execution of a job. Consequently, when a job fails, it can resume from the last checked point and run forward instead of restarting.

To run a big data loading job from Designer with recovery enabled:

1. In the project area, select the job name.
2. Right-click and choose *Execute*. The software prompts you to save any changes.
3. Make sure that the *Enable Recovery* option is selected in the *Execution Properties* window.

   > Note
   >
   > The *Enable Recovery* checkbox must be selected for the software to record the steps during the job. If the checkbox isn't selected and the job fails, you cannot recover the job. In this case, you have to perform any recovery operation manually and restart the job.

4. Check for the primary key in the target table of the data flow that contains the big data file source. If there is no primary key, then you have the option to either add the primary key column in the target table or generate the RowID column as the primary key column in the source file, and then pass it into the target tables.
   a) To add the primary key in the target table: If the original schema of the target table does not have a primary key, turn on the *Use Input Keys* in the target table's option and make sure that the input to the target table has a primary key. For example, the row ID column that you enter in the source file can be passed to the target table as a primary key.
   b) To generate Row ID column for the source file: Type a valid column name in the ROW ID input field located in the path *File Reader* > *General*.

      > Note
      >
      > A Row ID is automatically added into the schema as first column on the top. The column name is the name you typed, and the column Data Type is double.

A primary key column is necessary for fast recovery, as the engine uses auto correct loading for the last failed job. If the primary key column is not selected, then the auto correct loading will be very slow. To override this slow auto correct loading, you have the option to add the ROW ID column manually, if the target table does not have a primary key.

Data Services automatically turns on auto-correct loading for the recovery batch regardless of the target setting. Furthermore, for HANA, Netezza and Sybase IQ, the recovery batch auto-correct load uses bulk loading for better performance.

2.18.6.2 Limitations

There are some limitations that need to be met to allow a file to be loaded in recovery mode. The file along with the data flow and job contained in the file, needs to meet the following criteria:

- The file is a delimited file or fixed-width file
- The file has a row delimiter
- The job is running with *Enable recovery*
- The job is not a real-time job
- The job is not running in the Debugger
- The data flow is not within a continuous workflow
- The data flow is not audited
- The data flow is not an ABAP dataflow
- The data flow is not running in sub data flow mode
- The data flow does not contain any loader that does not connect to the file reader
- The data flow does not have the following transforms and functions:
  - Row_Generation
  - Date_Generation
  - gen_row_num
  - gen_row_num_by_group
  - CDC source
  - Target that is included in transaction
- The downstream of the file source does not have the following transforms:
  - Query transform that has an order by clause
  - Query transform that has a group by clause
  - Query transform that selects distinct rows
  - Query transform that has any aggregate function
  - Query transform that has a join in which file is an inner source

    > Note
    >
    > You can use the join rank to control the rank of the source relative to other tables and files, joined in a data flow. Sources with higher join ranks are joined before sources with lower join ranks.

  - Hierarchy_Flattening transform
  - Table_Comparison transform
  - Reverse_Pivot transform

    > Note
    >
    > Reverse_Pivot transform on the downstream of the big file is allowed, only when "input data is grouped" option is checked.

- ○ Validation transform
- ○ Merge

> i Note
>
> Merge transform on the downstream of the big file is allowed, only when all inputs of the Merge transform are directly or indirectly from the file.

- ○ Data Quality transform that uses a break key, such as Match transform
- ○ Either the target table has a primary key or the input to the target table has a primary key and the target’s 'Use input key' option is set to yes.

If any of the above conditions is not satisfied, recovery on the file is not enabled. However, there is no effect on the other source files or dataflows.

If the data flow contains multiple file sources, only one file source is chosen to run in recovery mode.

2.19 Capturing Changed Data

Related Information

Understanding changed-data capture [page 738]
Using CDC with Oracle sources [page 740]
Using CDC with Attunity mainframe sources [page 753]
Using CDC with Microsoft SQL Server databases [page 769]
Using CDC with timestamp-based sources [page 783]
Using CDC for targets [page 797]
Using CDC with SAP Sybase Replication Server [page 759]

2.19.1 Understanding changed-data capture

When you have a large amount of data to update regularly and a small amount of system down time for scheduled maintenance on a data warehouse, update data over time, or delta load. Two commonly used delta load methods are full refresh and changed-data capture (CDC).

2.19.1.1 Full refresh

Full refresh is easy to implement and easy to manage. This method ensures that no data will be overlooked or left out due to technical or programming errors. For an environment with a manageable amount of source data, full refresh is an easy method you can use to perform a delta load to a target system.

2.19.1.2 Capturing only changes

After an initial load is complete, you can choose to extract only new or modified data and update the target system. Identifying and loading only changed data is called changed-data capture (CDC). This includes only incremental data that has changed since the last refresh cycle. SAP Data Services acts as a mechanism to locate and extract only the incremental data that changed since the last refresh.

Improving performance and preserving history are the most important reasons for using changed-data capture.

- Performance improves because with less data to extract, transform, and load, the job typically takes less time.
- If the target system has to track the history of changes so that data can be correctly analyzed over time, the changed-data capture method can provide a record of these changes.
  For example, if a customer moves from one sales region to another, simply updating the customer record to reflect the new region negatively affects any analysis by region over time because the purchases made by that customer before the move are attributed to the new region.

This section discusses both general concepts and specific procedures for performing changed-data capture in the software.

2.19.1.3 Source-based and target-based CDC

Changed-data capture can be either source-based or target-based.

2.19.1.3.1 Source-based CDC

Source-based changed-data capture extracts only the changed rows from the source. It is sometimes called incremental extraction. This method is preferred because it improves performance by extracting the least number of rows.

SAP Data Services offers access to source-based changed data that various software vendors provide. The following table shows the supported data sources, changed-data capture products, and techniques.

Data Source	Products or techniques
Oracle 9i and later	Use Oracle's CDC packages to create and manage CDC tables. These packages make use of a publish and subscribe model. You can create a CDC datastore for Oracle sources using the Designer. You can also use the Designer to create CDC tables in Oracle, then import them for use in jobs.
Mainframe data sources (Adabas, DB2 UDB for z/OS, IMS, SQL/MP, VSAM, flat files) accessed with Attunity Connect	For mainframe data sources that use Attunity to connect to the software, you can use Attunity Streams 4.6.
Microsoft SQL Server databases	Use Microsoft SQL Replication Server to capture changed data from SQL Server databases.

Data Source	Products or techniques
Other sources	Use date and time fields to compare source-based changed-data capture job runs. This technique makes use of a creation and/or modification timestamp on every row. You can compare rows using the time of the last update as a reference. This method is called timestamp-based CDC.
SAP ODP sources i Note SAP extractors interface has been enhanced and replaced with OPD sources. Refer to the What's New Guide for Operational Data Provisioning [ODP] sources	Define CDC filtering criteria when importing an ODP source.
SAP Sybase Replication Server	Use SAP Sybase Replication Server to capture changed data.

Related Information

Using CDC with Oracle sources [page 740]
Using CDC with Attunity mainframe sources [page 753]
Using CDC with Microsoft SQL Server databases [page 769]
Using CDC with timestamp-based sources [page 783]
Using CDC with SAP Sybase Replication Server [page 759]
Supplement for SAP: Connecting to SAP Applications, Capturing changed data [page 2444]

2.19.1.3.2 Target-based CDC

Target-based changed-data capture extracts all the data from the source, but loads only the changed rows into the target.

Target-based changed-data capture is useful when you want to capture history but do not have the option to use source-based changed-data capture. The software offers table comparison to support this method.

2.19.2 Using CDC with Oracle sources

If your environment must keep large amounts of data current, the Oracle Change Data Capture (CDC) feature is a simple solution to limiting the number or rows that the software reads on a regular basis. A source that reads only the most recent operations (INSERTS, UPDATES, DELETES), allows you to design smaller, faster delta loads.

2.19.2.1 Overview of CDC for Oracle databases

With Oracle 9i or higher, SAP Data Services manages the CDC environment by accessing Oracle's Publish and Subscribe packages. Oracle publishes changed data from the original table to its CDC table.

The Designer allows you to create or import CDC tables and create subscriptions to access the data in the CDC table. Separate subscriptions allow each user to keep track of the last changed row that he or she accessed.

You can also enable check-points for subscriptions so that the software only reads the latest changes in the CDC table.

Oracle uses the following terms for Change Data Capture:

Term	Description
Change (CDC) table	A relational table that contains changed data that results from DML operations performed on a source table.
Change set	A group of CDC tables that are transactionally consistent. For example, SalesOrder and SalesItem tables should be in a change set to ensure that changes to an order and its line items are captured together.
Change source	The database that contains one or more change sets.
Publisher	The person who captures and publishes the changed data. The publisher is usually a database administrator (DBA) who creates and maintains the schema objects that make up the source database and staging database.
Publishing mode	Specifies when and how to capture the changed data. For details, see the following table of publishing modes.
Source database	The production database that contains the data that you extracted for your initial load. The source database contains the source tables.
Staging database	The database where the changed data is published. Depending on the publishing mode, the staging database can be the same as, or different from, the source database.
Subscriber	A user that can access the published data in the CDC tables.
Subscription	Controls access to the change data from one or more source tables within a single change set. A subscription contains one or more subscriber views.
Subscriber view	The changed data that the publisher has granted the subscriber access to use.

Oracle supports the following publishing modes:

Synchronous

- Data is captured using internal triggers on the source tables to store changes in CDC tables.
- Captured data is available in real-time.
- CDC tables must reside in the source database.
- Adds overhead to source database at capture time.
- Available in Oracle 9i and 10G.

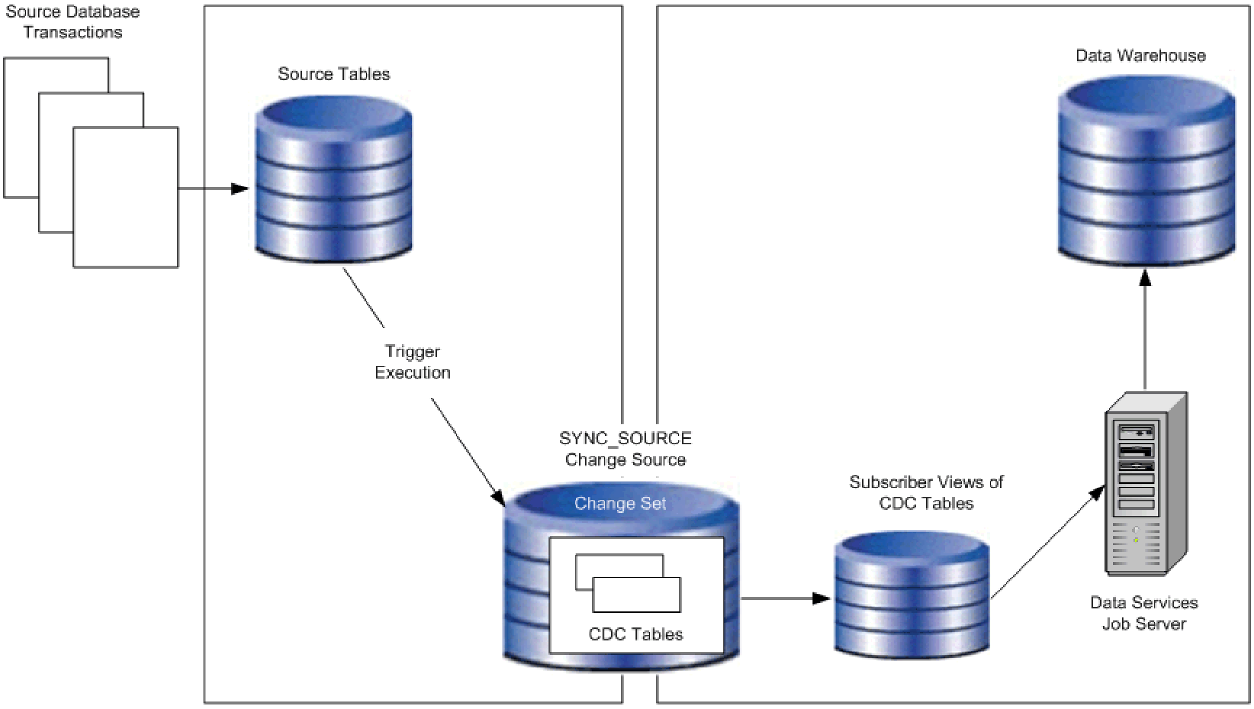
Asynchronous HotLog

- Data is captured using redo or archive logs from the source database.

	• Captured data is available in near real-time. • A change set contains multiple CDC tables and must reside locally in the source database. • Improves performance because data is captured offline. • Available in Oracle 10G only.
Asynchronous AutoLog	• Data is captured using redo logs managed by log transport services that automate transfer from source database to staging database. • Availability of captured data depends on the frequency of redo log switches on the source database. • A change set contains multiple CDC tables and can be remote or local to the source database. • Improves performance because data is captured offline. • Available in Oracle 10G only.

2.19.2.1.1 Oracle CDC in synchronous mode

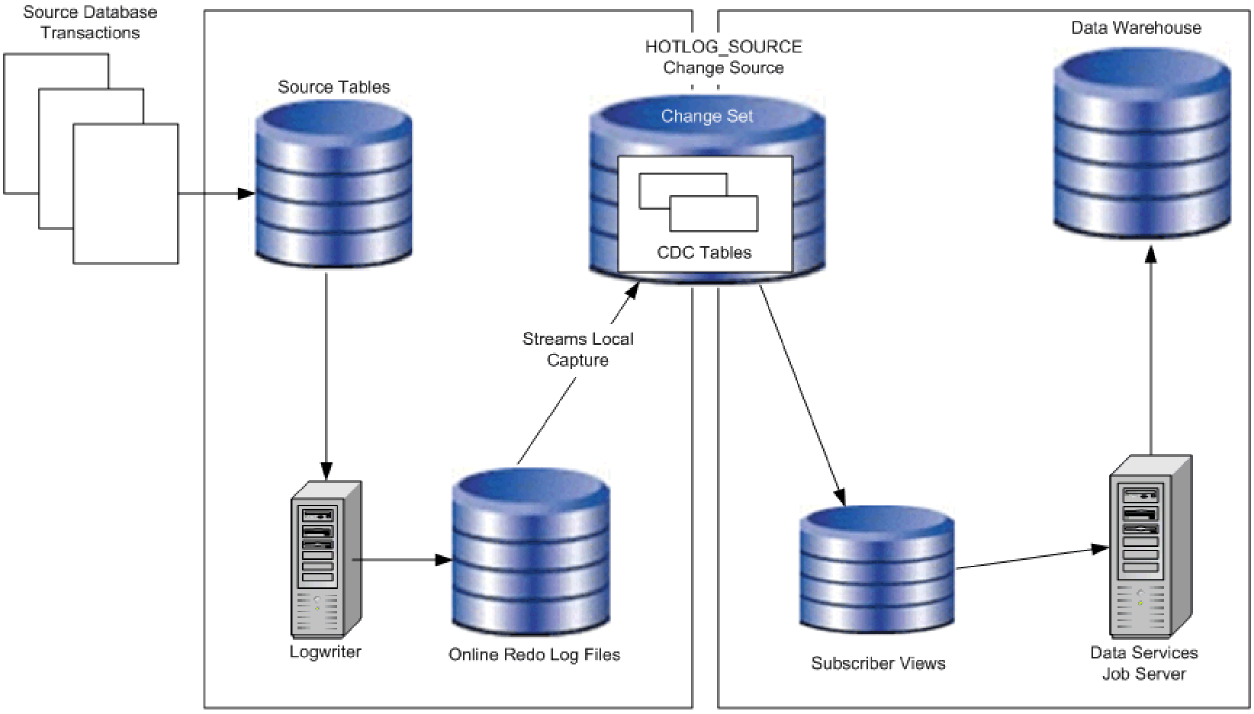
The following diagram shows how the changed data flows from Oracle CDC tables to SAP Data Services in synchronous mode.



When a transaction changes a source table, internal triggers capture the changed data and store it in the corresponding CDC table.

2.19.2.1.2 Oracle CDC in asynchronous HotLog mode

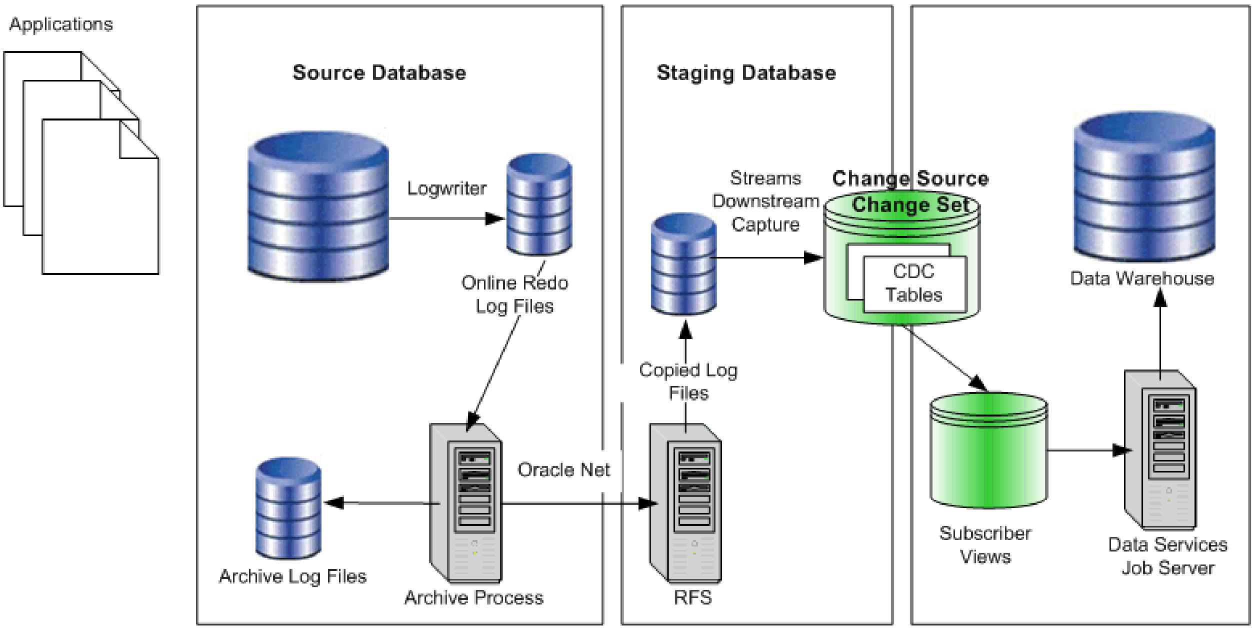
The following diagram shows how the changed data flows from Oracle CDC tables to SAP Data Services in asynchronous HotLog mode.



When a transaction changes a source table, the Logwriter records the changes in the Online Log Redo files. Oracle Streams processes automatically populate the CDC tables when transactions are committed.

2.19.2.1.3 Oracle CDC in asynchronous AutoLog mode

The following diagram shows how the changed data flows from Oracle CDC tables to SAP Data Services in asynchronous AutoLog mode.



When the log switches on the source database, Oracle archives the redo log file and copies the Online Log Redo files to the staging database. Oracle Streams processes populate the CDC tables from the copied log files.

Note

The Oracle archive process requires uninterrupted connectivity through Oracle Net to send the redo log files to the remote file server (RFS).

2.19.2.2 Setting up Oracle CDC

Use the following system requirements on your Oracle source database server to track changes:

- Install Oracle's CDC packages. These packages are installed by default. However, if a CDC package needs to be re-installed, open Oracle's `Admin` directory, then find and run Oracle's SQL script `initcdc.sql`.
  - Synchronous CDC is available with Oracle Standard Edition and Enterprise Edition.
  - Asynchronous CDC is available with Oracle Enterprise Edition only.
- Enable Java.
- Set source table owner privileges so CDC tables can be created, purged, and dropped as needed.
- Give datastore owners the SELECT privilege for CDC tables and the SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE privileges.
- For synchronous CDC, enable Oracle's system triggers.
- For asynchronous AutoLog CDC:
  - The source database DBA must build a LogMiner data dictionary to enable the log transport services to send this data dictionary to the staging database. Oracle automatically updates the data dictionary with any source table DDL operations that occur during CDC to keep the staging tables consistent with the source tables.

- ○ The source database DBA must also obtain the SCN value of the data dictionary build. If you will use the Designer to create CDC tables, you need to specify the SCN in the wizard.
- ○ The publisher (usually the source database DBA) must configure log transport services to copy the redo log files from the source database system to the staging database system and to automatically register the redo log files.

Related Information

To create Oracle CDC tables in the software [page 746]

2.19.2.3 To create a CDC datastore for Oracle

To access CDC tables, create a CDC datastore using the Designer. A CDC datastore is a read-only datastore that can only access tables. Like other datastores, you can create, edit, and access a CDC datastore from the *Datastores* tab of the object library.

1. Create a database datastore with the *Database Type* option set to `Oracle`.
2. Select the *CDC* check box.
3. Select an *Oracle version*.

   The Designer only allows you to select the Oracle versions that support CDC packages.
4. Specify the name of your staging database (the change source database where the changed data is published) in *Connection name*.
5. Enter the *User* and *Password* for your staging database and click *OK*.

   You can use this datastore to browse and import CDC tables.

2.19.2.4 Importing CDC data from Oracle

You must create a CDC table in Oracle for every source table you want to read from before you can import that CDC table using SAP Data Services. Use one of the following ways:

- Use an Oracle utility to create CDC tables
- Use the Designer to create CDC tables

2.19.2.4.1 To use existing Oracle CDC tables

1. Import an Oracle CDC table by right-clicking the CDC datastore name in the object library and selecting *Open*, *Import by Name*, or *Search*.

   If you select Open, you can browse the datastore for existing CDC tables using the Datastore Explorer.

2. When you find the table that you want to import, right-click it and select *Import*.

2.19.2.4.2 To create Oracle CDC tables in the software

The software provides the ability to create Oracle CDC tables for all publishing modes:

- Synchronous CDC
- Asynchronous HotLog CDC
- Asynchronous AutoLog CDC

1. In the object library, right-click a CDC datastore and select *Open*.
2. In the Datastore Explorer, right-click the white space in the *External Metadata* section, and select *New*.

   The New CDC table wizard opens. This wizard allows you to add a CDC table.

   > Note
   >
   > If the Datastore Explorer opens and no CDC tables exist in your datastore, this wizard opens automatically.

3. Select the publishing mode on the first page of the wizard.

   If your source database is Oracle 9i, you can only select the Synchronous mode. The Asynchronous modes are disabled.

   If your source database is Oracle 10G, the wizard selects the Asynchronous HotLog mode by default.

   If your source database uses Asynchronous AutoLog publishing mode, select Asynchronous AutoLog and provide the following source database connection information:

Field	Description
Connection name	The name of the database where the Change Source resides. Use the service name of the Oracle Net service configuration.
User Name	The user name for the source database DBA.
Password	The password for the Change Source user.

4. Click *Next*. The second page of the wizard appears.
5. Specify the source table information in the second page of the wizard.
   a) Click the *Search* button to see a list of non-CDC external tables available in this datastore. To filter a search, enter values for a table Owner and/or Name. You can use a wild-card character (%) to perform pattern matching for Name or Owner values.
   b) (Optional) Select *Generate before-images* if you want to track before- and after-images in the new CDC table.
   c) Click a name in the list of returned tables and click *Next* to create a CDC table using the selected table as a source table.
6. Specify the *CDC table owner* for the new CDC table.

   By default, the owner name of the new CDC table is the owner name of the datastore. The source table owner name is also displayed in the CDC table owner list box. If the owner name you want to use is not in the list, enter a different owner name.

7. Specify the *CDC table name* for the new CDC table.

   By default, the software generates a table name using the following convention: CDC__SourceTableName.
8. By default, all columns are selected. Specify which columns to include or exclude from the CDC table in one of the following ways: Either remove the check mark from the box next to the name of each column that you want to exclude, or click *Unselect All* and place a check mark next to the name of each column that you want to include.
9. For synchronous publishing mode:
   a) Click *Finish*. The Designer connects to the Oracle instance, creates the CDC table on the Oracle server, and imports the table's metadata into the repository. All tables that the software imports through a CDC datastore contain a column that indicates which operation to perform for each row. For an Oracle CDC table, this column is called Operation$. In addition to this column, Oracle adds other columns when it creates a CDC table. These columns all use a dollar sign as a suffix.
   b) Click *OK* on the information dialog. This dialog confirms that Oracle created a new CDC table, then imported it successfully into the software.
10. For asynchronous (HotLog or AutoLog) publishing mode, click *Next*.
11. For asynchronous HotLog publishing mode, specify the change set information in the fourth page of the wizard.
    a) If you would like to add this change table to an existing change set to keep the changes transactionally consistent with the tables in the change set, select a name from the drop-down list for *Change set name*. Alternatively, you can create a new change set by typing in the name.
    b) Select *Stop capture on DDL* if a DDL error occurs and you do not want to capture data.
    c) Select *Define retention period* to enable the *Begin Date* and *End Date* text boxes.
    d) Click *Finish*.
    The Designer connects to the Oracle instance, creates the CDC table on the Oracle server, and imports the table's metadata into the software's repository. All tables that the software imports through a CDC datastore contain a column that indicates which operation to perform for each row. For an Oracle CDC table, this column is called Operation$. In addition to this column, Oracle adds other columns when it creates a CDC table. These columns all use a dollar sign as a suffix.
12. For asynchronous AutoLog publishing mode, specify the change set and change source information in the fourth page of the wizard.
    a) If you would like to add this change table to an existing change set to keep the changes transactionally consistent with the tables in the change set, select a name from the drop-down list for *Change set name*. Alternatively, you can create a new change set by typing in the name.
    b) If you would like to add this change table to an existing change source, select a name from the drop-down list for *Change source name*.
    c) If you want to create a new change source, type the name of the CDC change source and the name of the source database. You can obtain this name from the source database Global_Name table SCN value of the data dictionary build.
    d) Select *Stop capture on DDL* if a DDL error occurs during data capture and you do not want to capture data.
    e) Select *Define retention period* to enable the *Begin Date* and *End Date* text boxes.
    f) Click *Finish*.
    The Designer connects to the Oracle staging database, creates the CDC table on the change source, and imports the table's metadata into the software's repository. All tables that the software imports through a CDC datastore contain a column that indicates which operation to perform for each row. For an Oracle

CDC table, this column is called Operation$. In addition to this column, Oracle adds other columns when it creates a CDC table. These columns all use a dollar sign as a suffix.

Related Information

Using before-images [page 750]

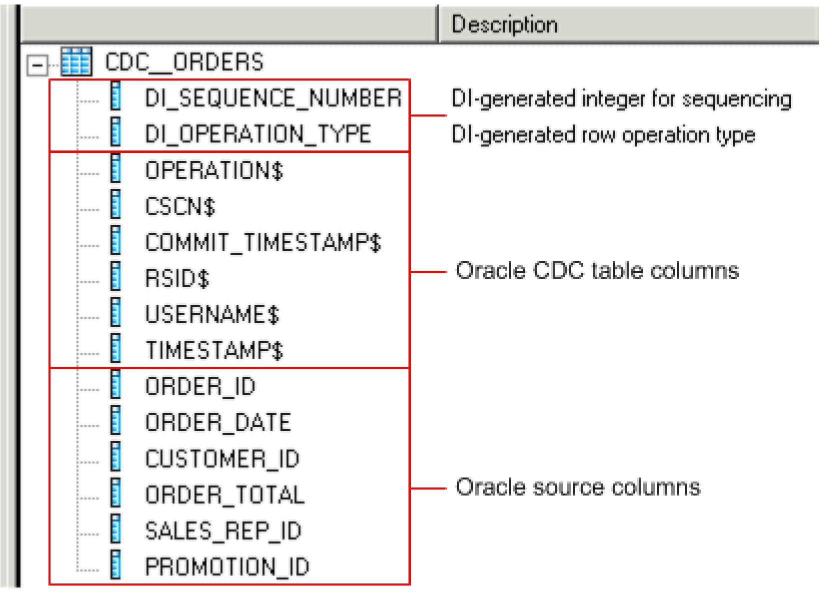
2.19.2.5 Viewing an imported CDC table

To view an imported CDC table:

1. Find your CDC datastore in the object library.
2. Expand the *Tables* folder.
3. Double-click a table name or right-click and select *Open*.

When the software imports a CDC table, it also adds two columns to the table's schema: `DI_SEQUENCE_NUMBER` with the data type integer and `DI_OPERATION_TYPE` with the data type varchar(1).

An imported Oracle CDC table schema looks like the following:



Description

CDC__ORDERS

DI_SEQUENCE_NUMBER — DI-generated integer for sequencing

DI_OPERATION_TYPE — DI-generated row operation type

OPERATION$
CSCN$
COMMIT_TIMESTAMP$
RSID$
USERNAME$
TIMESTAMP$

Oracle CDC table columns

ORDER_ID
ORDER_DATE
CUSTOMER_ID
ORDER_TOTAL
SALES_REP_ID
PROMOTION_ID

Oracle source columns

This example has eight control columns added to the original table:

- Two generated by the software
  - `DI_SEQUENCE_NUMBER` - Contains an integer for sequencing.
  - `DI_OPERATION_TYPE` - Contains the row operation type.
- Six Oracle control columns: OPERATION$, CSCN$, COMMIT_TIMESTAMP$, RSID$, USERNAME$, TIMESTAMP$

ℹ Note

The Oracle control columns vary, depending on the options that were selected when the CDC table is created. All Oracle control columns end with a dollar sign ($).

Related Information

The DI_SEQUENCE_NUMBER column [page 749]
The DI_OPERATION_TYPE column [page 749]

2.19.2.5.1 The DI_SEQUENCE_NUMBER column

The `DI_SEQUENCE_NUMBER` column starts with zero at the beginning of each extraction. This field increments by one each time the software reads a row except when it encounters a pair of before- and after-images for an UPDATE operation. Both the before- and after-images receive the same sequence number. This sequencing column provides a way to collate image pairs if they are separated as a result of the data flow design.

Related Information

Using before-images [page 750]

2.19.2.5.2 The DI_OPERATION_TYPE column

The possible values for the `DI_OPERATION_TYPE` column are:

- `I` for INSERT
- `D` for DELETE
- `B` for before-image of an UPDATE
- `U` for after-image of an UPDATE

When the software reads rows from Oracle, it checks the values in column `Operation$` and translates them to the software values in the `DI_OPERATION_TYPE` column.

Operation$	DI_OPERATION_TYPE
I	I
D	D
UO, UU	B

Operation$	DI_OPERATION_TYPE
UN	U

2.19.2.6 To configure an Oracle CDC source table

When you drag a CDC datastore table into a data flow, it automatically becomes a source object.

1. Drag a CDC datastore table into a data flow.
2. Click the name of this source object to open its Source Table Editor.
3. Click the CDC Options tab.
4. Specify a value for the *CDC subscription name*.

   For more information, see the *Reference Guide*.

2.19.2.6.1 Using check-points

When a job in SAP Data Services runs with check-pointing enabled, software uses the source table's subscription name to read the most recent set of appended rows. If you do not enable check-pointing, then the job reads all the rows in the table and increases processing time.

To use check-points, enter a name in the *CDC Subscription* name box on the Source Table Editor and select the *Enable check-point* option.

> i Note
>
> To avoid data corruption problems, do not reuse data flows that use CDC datastores because each time a source table extracts data it uses the same subscription name. This means that identical jobs, depending upon when they run, can get different results and leave check-points in different locations in the table. When you migrate CDC jobs from test to production, for example, a best practice scenario would be to change the subscription name for the production job so that the test job, if ever runs again, will not affect the production job's results.

2.19.2.6.2 Using before-images

If you want to retrieve the before-images of UPDATE rows, prior to when the update operation is applied to the target, the software can expand the UPDATE row into two rows: one row for the before-image of the update, and one row for the after-image of the update. The before image of an update row is the image of the row before the row is changed, and the after image of an update row refers to the image of the row after the change is applied.

The default behavior is that a CDC reader retrieves after-images only. By not retrieving before-images, fewer rows pass through the engine which allows the job to execute in less time.

You can use before-images to:

- Update primary keys

> ℹ Note
>
> Under most circumstances, when source tables are updated, their primary keys do not need to be updated.

- Calculate change logic between data in columns
  For example, you can calculate the difference between an employee's new and old salary by looking at the difference between the values in salary fields.

2.19.2.6.2.1 To capture before-images for update rows:

1. At CDC table creation time, make sure the Oracle CDC table is also setup to retrieve full before-images.

   If you create an Oracle CDC table using the Designer, you can select the *Generate before-images* to do this.
2. Select *Get before-images for each update row* in the CDC table's source editor.

   If the underlying, CDC table is not set-up properly, enabling the Get before-images for each update row option has no effect.

Once you select the *Get before-images for each update row* option, for every update, the software processes two rows. In addition to the performance impact of this data volume increase, the before- and after-image pairs may be separated or lost depending on the design of your data flow. This would cause data integrity issues.

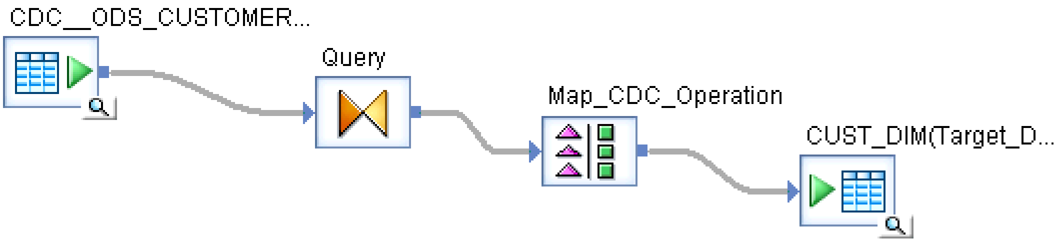
The Map_CDC_Operation transform can resolve problems, but undesirable results may still occur due to programming errors. When using functions and transforms that re-order, re-direct, eliminate, and multiply the number of rows in a data flow (for example, due to the use of the group by or order by clauses in a query) be aware of the possible impact to targets.

2.19.2.7 To create a data flow with an Oracle CDC source

To use an Oracle CDC source, you use a Query transform to remove the Oracle control columns and the Map_CDC_Operation transform to interpret the control columns and take appropriate actions.

1. In the *Designer Object Library* pane, drag the Oracle CDC table, Query, and Map_CDC_Operation transforms to the data flow workspace area.

   > ℹ Note
   >
   > A data flow can contain only one CDC source.
2. Configure the CDC table.
3. Add the appropriate target table and connect the objects.



4. In the *Project Area* pane, double-click a transform.
   The *Query Editor* opens, displaying the transform.
5. Map the *Data Services* control columns and the source table columns that you want in your target table.

   The Map_CDC_Operation transform uses the values in the column in the *Row Operation Column* box to perform the appropriate operation on the source row for the target table. For an Oracle CDC source table, the `DI_OPERATION_TYPE` column is automatically selected as the Row operation column.

   The operations can be `INSERT`, `DELETE`, or `UPDATE`. For example, if the operation is `DELETE`, the corresponding row is deleted from the target table.

Related Information

To configure an Oracle CDC source table [page 750]
Reference Guide: Transforms [page 1067]

2.19.2.8 Maintaining CDC tables and subscriptions

2.19.2.8.1 Purging CDC tables

Periodically purge CDC tables so they do not grow indefinitely.

> Note
>
> The software does not provide this functionality. Refer to your Oracle documentation for how to purge data that is no longer being used by any subscribers.

2.19.2.8.2 To drop Oracle CDC subscriptions or tables

Oracle's purge facility does not purge any data that has not been read by all subscriptions. As a result, it is a good practice to drop any subscriptions that are no longer needed. You can drop Oracle CDC tables and their subscriptions from the *Datastore Explorer* window in the Designer.

1. In the *Local Object Library* pane, right-click a CDC datastore, and click *Open*.
2. In the *Datastore Explorer* window, click *Repository Metadata*.
3. Right-click a table, point to *CDC maintenance*, and do one of the following:
   - Point to *Drop subscription*.
     - *Drop Subscription* opens the list of subscriptions you created in the software for the selected table. Oracle subscriptions are associated with these subscription names. Select each subscription name to drop it from Oracle and delete it from the repository.
   - Click *Drop table*.
     - This option drops the Oracle CDC table and also deletes it from the repository.

2.19.2.9 Limitations

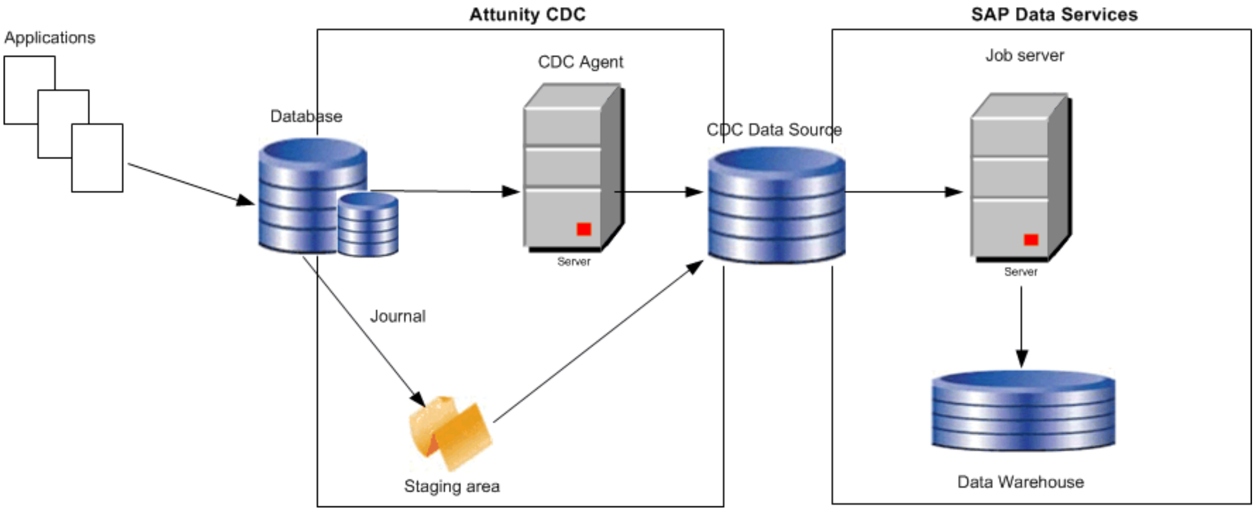
The following limitations exist when using CDC with Oracle sources:

- You cannot use the following transforms and functions with a source table imported with a CDC datastore because of the existence of the SAP Data Services generated columns for CDC tables. The software cannot compare or search these columns.
  - Table_Comparison, Key_Generation, and SQL transforms
  - All database functions, such as `lookup`, `lookup_ext`, key_generation, sql, and total_rows
- You can only create one CDC source in a data flow.
- Oracle CDC captures DML statements, including `INSERT`, `DELETE`, and `UPDATE`. However, Oracle CDC does not support the following operations because they disable all database triggers:
  - Direct-path `INSERT` statements
  - The `multi_table_insert` statement in parallel DML mode
- If you are using check-pointing and running your job in recovery mode, the recovered job will begin to review the job at the start of the CDC table. Check-points are ignored.

2.19.3 Using CDC with Attunity mainframe sources

If your environment must keep large amounts of data current, the mainframe CDC feature is a simple solution to limiting the number of rows that must be read on a regular basis. A source that reads only the most recent operations (INSERTS, UPDATES, DELETES) allows you to design smaller, faster delta loads.

SAP Data Services captures changed data on Attunity mainframe data sources and applies it to a target system. The following diagram shows the path that the data takes from Attunity CDC to SAP Data Services.



- The Attunity CDC Agent monitors the database journal for changes to specific tables. After the first request to capture changes, the CDC agent stores a context that the agent uses as a marker to not recapture changes prior to it.
- The CDC Agent sends the changed data to an optional staging area. The advantages of a staging area are:
  - A single journal scan can extract changes to more than one table. Without a staging area, multiple journal scans, one for each changed table, is required to extract changes.
  - Extracts only committed changes which is less processing than extracting every change. Less processing also occurs during recovery of a failed job because the recovery process does not need to back out the uncommitted changes.

  However, a staging area requires additional storage and processing overhead. Refer to the Attunity CDC documentation for details.
- Attunity Connect CDC sends the changes to the CDC data sources through which the software can access the changes using standard ODBC or JDBC.

2.19.3.1 Setting up Attunity CDC

If you currently use Attunity as the connection to SAP Data Services to extract data from mainframe sources, create an Attunity CDC data source in Attunity Studio. The following steps summarize the procedure for using the Attunity Studio wizard to create a CDC data source.

- Specify your data source.
- Based on your data source, choose one of the following methods to capture changes and specify the location of the journal:
  - VSAM under CICS—By CICS Log stream
  - DB2 on OS/390 and z/OS platforms—By DB2 Journal
  - DB2 on OS/400—By DB400 Journal
  - DISAM on Windows—By Journal

  For a complete list of supported data sources, see the Attunity Connect CDC document.
- Select a name for your CDC agent.

- Specify if you want to capture before images for update operations. If you do not specify this option in Attunity Studio, you will not capture before images even if you specify the option *Get before-image for each update row*.
- Select the tables to monitor for changes.

The Attunity Studio wizard generates the following components that you need to specify on the Datastore Editor when you define an Attunity CDC datastore:

- A CDC data source name that you specify in the option *Data source*. Attunity generates the CDC data source on the same computer as the CDC agent by default. You have the option of placing the CDC data source on the client (same computer as SAP Data Services). Obtain the host name of this computer to specify in the option *Host location*.
- A workspace for the CDC agent to manage the change capture event queue. You specify the workspace name in the option *Attunity workspace*.

For more information, refer to the CDC setup section in the *Attunity Connect: The Change Data Capture Solution*.

2.19.3.2 Setting up the software for CDC on mainframe sources

To use SAP Data Services to read and load changed data from mainframe sources using Attunity, do the following procedures on the Designer:

- Create a CDC datastore for Attunity
- Import metadata for Attunity tables
- Configure a mainframe CDC source
- Build real-time jobs using metadata

2.19.3.2.1 Creating CDC datastores

The CDC datastore option is available for all mainframe interfaces to SAP Data Services.

Related Information

Datastores [page 209]

2.19.3.2.1.1 To create a CDC datastore for Attunity

1. Open the Datastore Editor.
2. Enter a name for the datastore.

3. In the *Datastore type* box, select Database.
4. In the *Database type* box, select Attunity_Connector.
5. Check the *Enable CDC* box to enable the CDC feature. You can enable CDC for the following data sources. For the current list of data sources, refer to the Attunity web site.
   - VSAM under CICS
   - DB2 UDB for z/OS
   - DB2 UDB for OS/400
6. In the *Data source* box, specify the name of the Attunity CDC data source. You can specify more than one data source for one datastore, but you cannot join two CDC tables. You might want to specify multiple data sources in one Attunity datastore for easier management. If you can access all of the CDC tables through one Attunity data source, it is easier to create one datastore, enter the connection information once, and import the tables.

   If you list multiple data source names for one Attunity Connector datastore, ensure that you meet the following requirements:
   - Do not specify regular Attunity data sources with CDC data sources in the same datastore. The software imports data from regular Attunity data sources differently than from CDC data sources.
   - All Attunity data sources must be accessible by the same user name and password.
   - All Attunity data sources must use the same workspace. When you setup access to the data sources in Attunity Studio, use the same workspace name for each data source.
7. In the *Host location* box, specify the name of the host on which the Attunity data source daemon exists.
8. In the *Port* box, specify the Attunity daemon port number. The default value is 2551.
9. Specify the Attunity server workspace name that the CDC agent uses to manage the change capture event queue for the CDC data source.
10. Complete the rest of the dialog and click *OK*.

    You can now use the new datastore connection to import metadata tables into the current repository.

Once saved, this datastore becomes a CDC datastore.

2.19.3.3 Importing mainframe CDC data

After you create a CDC datastore, you can use it to import CDC table metadata. In the object library, right-click the datastore name and select *Open*, *Import by Name*, or *Search*. For mainframe CDC, only the CDC tables that you selected in the procedure *Setting up Attunity CDC* [page 754] are visible when you browse external metadata. Functions and templates are not available because the Attunity CDC datastore is read-only.

The SAP Data Services import operation adds the following columns to the original table:

Column name	Data type	Source of column
DI_SEQUENCE_NUMBER	integer	Generated by SAP Data Services
DI_OPERATION_TYPE	varchar(1)	Generated by SAP Data Services
Context	varchar(128)	Supplied by Attunity Streams

Column name	Data type	Source of column
Timestamp	varchar(26)	Supplied by Attunity Streams
TransactionID	varchar(4)	Supplied by Attunity Streams
Operation	varchar(12)	Supplied by Attunity Streams
tableName	varchar(256)	Supplied by Attunity Streams

2.19.3.3.1 The DI_SEQUENCE_NUMBER column

The DI_SEQUENCE_NUMBER column starts with zero at the beginning of each extraction. This field increments by one each time the software reads a row except when it encounters a pair of before- and after-images. Both the before- and after-images receive the same sequence number. This sequencing column provides a way to collate image pairs if they become separated as a result of the data flow design.

You can configure Attunity Streams to retrieve before- images of UPDATE rows before the software applies the UPDATE operation to the target. Note that if you do not configure Attunity Streams to capture before- images in the database, the software will discard the rows. For information about when to consider using before-images, see *Using before-images* [page 750].

If during the course of a data flow the before- and after-images become separated or get multiplied into many rows (for example, using GROUP BY or ORDER BY clauses in a query), you can lose row order.

The Map_CDC_Operation transform allows you to restore the original ordering of image pairs by using the DI_SEQUENCE_NUMBER column as its *Sequencing column*.

Related Information

Reference Guide: Transforms [page 1067]

2.19.3.3.2 The DI_OPERATION_TYPE column

SAP Data Services generates values in the DI_OPERATION_TYPE column. Valid values for this column are:

- I for INSERT
- D for DELETE
- B for before-image of an UPDATE
- U for after-image of an UPDATE

2.19.3.4 Configuring a mainframe CDC source

When you drag a CDC datastore table into a data flow, it automatically becomes a source object.

2.19.3.4.1 To configure a mainframe CDC table

1. Drag a CDC datastore table into a data flow.

   The table automatically becomes a source object.
2. Click the name of this source object to open its Source Table Editor.
3. Click the CDC Options tab.
4. Specify a value for the *CDC subscription name*.

   For more information, see the *Reference Guide*.

Related Information

Using mainframe check-points [page 758]
Using before-images [page 750]

2.19.3.5 Using mainframe check-points

Attunity CDC agents read mainframe sources and load changed data either into a staging area or directly into the CDC data source. Rows of changed data append to the previous load in the CDC data source.

When you enable check-points, a CDC job in SAP Data Services uses the subscription name to read the most recent set of appended rows and to mark the end of the read. If check-points are not enabled, the CDC job reads all the rows in the Attunity CDC data source and processing time increases.

To use check-points, on the Source Table Editor enter the *CDC Subscription name* and select the *Enable check-point* option.

If you enable check-points and you run your CDC job in recovery mode, the recovered job begins to review the CDC data source at the last check-point.

> Note
>
> To avoid data corruption problems, do not reuse data flows that use CDC datastores because each time a source table extracts data it uses the same subscription name. This means that identical jobs, depending upon when they run, can get different results and leave check-points in different locations in the file. When you migrate CDC jobs from test to production, a best-practice scenario is to change the subscription name for the production job. Therefore, if the test job ever runs again, it does not affect the production job's results.

2.19.3.5.1 Using before-images from mainframe sources

When you must capture before-image update rows.

1. Make sure Attunity Streams is set up to retrieve full before-images.
2. Select the *Get before-images for each update row* option in the CDC table's source editor.

   The underlying, log-based CDC capture software must be set up properly, otherwise enabling the Get before-images for each update row option in the software has no effect.

After you check the *Get before-images for each update row* option, the software processes two rows for every update. In addition to the performance impact of this data volume increase, the before- and after-image pairs could be separated or lost depending on the design of your data flow, which would cause data integrity issues.

The Map_CDC_Operation transform can resolve problems, but undesirable results can still occur due to programming errors. When you use functions and transforms that re-order, re-direct, eliminate, and multiply the number of rows in a data flow, be aware of the possible impact to targets.

Related Information

Using before-images [page 750]
Reference Guide: Transforms [page 1067]

2.19.3.6 Limitations

The following limitations exist for this feature:

- You cannot use the following transforms and functions with a source table imported with a CDC datastore because of the existence of the SAP Data Services generated columns for CDC tables. The software cannot compare or search these columns.
  - Table_Comparison, Key_Generation, and SQL transforms
  - All database functions, such as `lookup`, `lookup_ext`, key_generation, sql, and total_rows
- You can only create one CDC source in a data flow.

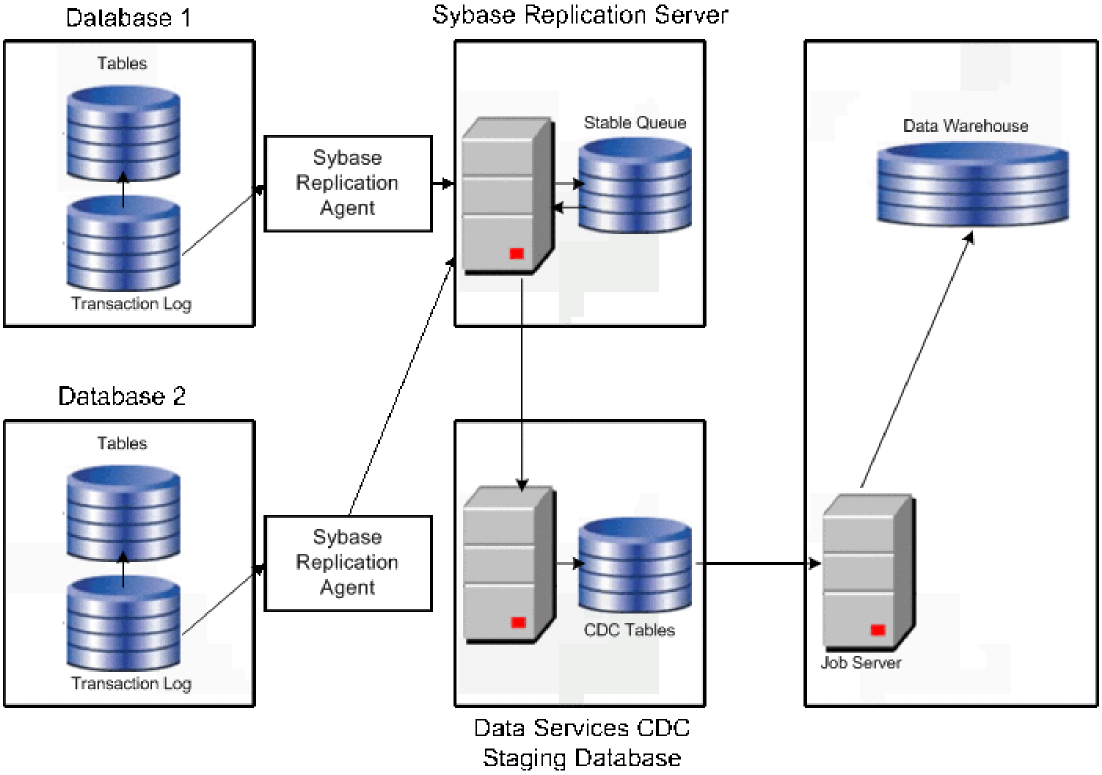
2.19.4 Using CDC with SAP Sybase Replication Server

If your environment must keep large amounts of data current with minimal impact on operational systems, then Data Services CDC feature combined with SAP Sybase Replication Server's replication functionality provides a mechanism to capture, transform, and propagate high volumes of data to data warehouse in real-time. This helps make rapid decisions, thereby keeping pace with the ever-changing environment.

SAP Sybase Replication Server CDC source in data flow reads only the most recent operations (INSERTS, UPDATES, DELETES), allowing you to design smaller, faster delta loads. Data Services

2.19.4.1 Overview of CDC for Sybase Replication Server

Data Services manages the CDC environment by building the CDC staging database and creating SAP Sybase Replication Server's replication definitions and subscriptions, using SAP Sybase Power Designer's Data Movement Model. Upon deployment of the replication definitions and subscriptions and the CDC staging database definitions, SAP Sybase Replication Server publishes changed data from the source table to its corresponding CDC table that resides in the CDC staging database. The CDC Table Source in Data Services' data flow reads the most recent change data from the last check-point from the CDC table and applies it to the target systems, after transformation. The following diagram shows how the changed data flows from the Replication Server to Data Services.



2.19.4.2 Setting up SAP Sybase Replication Server CDC Environment

You need to install the list of prerequisites from the table below for SAP Sybase Replication Server CDC. For more information about SAP Sybase components, see SAP Sybase documentation links.

Design time component	Function	Reference
SAP Sybase Power Designer 16.5	Modelling tool for creating the replication definitions, the subscriptions for Replication Server and the Data Services CDC staging database definitions. You need this tool along with Data Services Designer to model	*http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocen-*

Design time component	Function	Reference
	the source system to the target system CDC data flow.	*ter.help.pd.16.5/doc/html/title.html*
Data Services Designer 4.2	Designing the data flow for transforming change data before applying to the target system.	

Run-time component	Function	Reference
SAP Sybase Replication server 15.7.1	Capturing change data from the source database and publishing to the Data Services CDC staging database.	*http://infocenter.sybase.com/help/topic/com.sybase.infocenter.help.rs.15.7.1/title.htm*
isql tool	Deploying replication definitions and subscriptions in the Replication Server, enabling the source tables in the source databases for replication and creating the Data Services CDC database definitions in staging database.	*Utility Guide of Adaptive Server Enterprise*
SAP Sybase ASE 15.7.1	Staging Data Services CDC change data.	*http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.help.ase.15.7/title.htm*
SAP Sybase ASE 15.7.1 Job Scheduler	Purging the old change data from the CDC staging database (part of CDC environment management).	i Note You need to install Job Scheduler Agent as part of SAP Sybase ASE 15.7.1.
Data Services Job Server 4.2	Transforming the change data and applying to the Target databases.	

2.19.4.3 Modeling of Replication Process for Data Services CDC Staging Database

Power Designer automates the process of configuring SAP Sybase Replication Server, which includes marking the source table as replicated, creating the replication definitions and subscriptions and defining the Data Services CDC staging database.

> Note
>
> Tables in source database that needs to be replicated, must have primary key.

The following steps summarize the procedure to model the replication process for Data Services CDC Staging database in Power Designer:

1. Drop the `SybaseRepServerDataServicesCDC100.xem` file from Data Services install location `%`**`<DS_COMMON_DIR>`**`%/ext/cdc` into Power Designer's *Resource Files* > *Extended Model Definitions* folder.

   > Note
   >
   > Data Services Power Designer Extension is a part of the Data Services Package.

2. Create a new Data Movement Model in the Power Designer by selecting *Model* > *New* > *Data Movement Model*. Enter the *Model name* and in *Extesnions* choose *Replication Server 15.5* from the drop down list.
3. Attach *SAP Data Services CDC 14.2.0 extension* to it. To do this, go to *model* > *extensions* > *List of Extensions*. Click the attach an extension icon and choose *SAP Data Services CDC 14.2.0 extension*.

   > Note
   >
   > This extension works only with Replication Server 15.7 data movement modeling.

4. Run the *Replication Wizard* in the Power Designer by selecting *Tools* > *Replication Wizard*. It creates the replication server process, the source physical data model and the replicated physical data model.

   > Note
   >
   > The Replication Wizard reverse-engineers the source databases to import database objects into the data movement model. Choose only tables and following table properties: Primary Keys, Foreign Keys, Alternate Keys and indexes to create the replicated physical database model.

5. When you are on the *Publication Selection* window, in the *Replication Wizard* dialog box, select *Create or select a single publication to gather replicated tables* and choose replication type as *replication definitions*.

   > Note
   >
   > When the Replication Wizard finishes, Power Designer creates default replication definitions (called Articles) and the Physical Data Model for the Remote database with all selected tables from the source Physical Data Model and corresponding table columns and properties.

6. In the *Replication Wizard*, when you are on the *Remote Physical Data Model Selection* screen, choose the *DBMS* as *Sybase AS Enterprise 15.7*.
7. Save the Data Movement Model and Physical Data Model.
8. Select the Replication Process icon in the Data Movement Model diagram and choose *Add Data Services CDC Definitions* from context menu to run Data Services extension to create the following:
   a) CDC replication definitions
   b) CDC staging tables
   c) CDC system tables (DS_CDC_TABLES_MAP, DS_CHANGE_RETENTION and DS_PURGE_TIMESTAMP)

d) User SAPDS

ℹ Note

Selecting the *Add Data Services CDC Definitions* option automatically changes owner of the replicated tables, the purge stored procedure, the CDC staging table and the CDC system tables to SAPDS. It defines the CDC staging table name same as the Remote Table name of the article with suffix "_SAPCDC". It creates or overrides function strings: rs_delete, rs_insert, rs_update and rs_get_textptr for blob columns. The CDC staging table defines additional columns in addition to the RemoteTable columns to store the Replication Server system variables:

Function string	What it does
DS_SRCDB	stores system variable “rs_origin_db”
DS_SRCDB_SRV	stores system variable “rs_origin_ds”
DS_SRCDB_ID	stores system variable “rs_origin”
DS_SRCDB_COMMIT_TIME	stores system variable “rs_origin_commit_time”
DS_SRCDB_ORIGIN_QID	stores “rs_origin_qid”
DS_SRCDB_OP_TYPE	stores change operation types“I”- insert , “B”- Before image of update,“U”- After image of update, “D”-Delete
DS_CDCDB_PUBLISH_TIME	stores publish datetime of CDC database when record get inserted by the Replication Server

Add Data Services CDC Definitions command also sets “CDC Staging Table” attributes with the CDC staging table name for each article.

Additionally, *SAP Data Services CDC 14.2.0* extension adds two extended attributes, "Retention Period" and "CDC Staging Table" to Article object. Default value "Retention Period" extended attributes is 14 days. You can change this value.

2.19.4.4 Configuring Data Services Sybase Replication Server CDC environment

There are five steps to configure the SAP Data Services Sybase Replication Server CDC environment. These are:

1. Creating Data Services CDC staging database in the SAP Sybase ASE server.
2. Configuring the Replication Server to add the source databases and the CDC staging database.
3. Configuring the Data Services CDC staging database runtime environment in SAP Sybase ASE Server.
4. Deploying the replication definitions and subscriptions into the Replication Server.
5. Configuring the primary databases runtime environment.

2.19.4.4.1 Creating Data Services CDC staging database in SAP Sybase ASE Server

The following procedure describes how to create Data Services CDC staging database in SAP Sybase ASE server

1. Log in as database administrator and create a database that matches the name of Data Services CDC staging database, as defined in Power Designer's data movement model.
2. Create SAPDS log-in user. Lock this user to prevent log-in.

   > **Note**
   >
   > This user is reserved for Data Services CDC and should not be used for any other purpose. All the Data Services staging database objects are owned by user SAPDS.

3. Select the Data Services staging database Physical Data Model, in Power Designer's Data Movement Model and right-click the context to select *Generate Database* to generate the CDC database DDL script.

   > **Note**
   >
   > This automatically generates DDL of all the database objects that are required for Data Services CDC Staging database.

4. Check the CDC database DDL script that is generated as a file.
5. Deploy this DDL script using the isql tool. Enter the code:

   ```
   isql-Usa-Ppwd -Spds -Dpdb -i DS_CDC_staging_DDL_script.sql-
   oDS_CDC_staging_DDL_script.out
   ```

   where:

   - sa is the System Administrator user login on the DS CDC Staging data server.
   - pwd is the password for the System Administrator user login.
   - pds is the name of the DS CDC Staging data server.
   - pdb is the name of the DS CDC staging database.

2.19.4.4.2 Configuring Replication Server to add Source databases and CDC staging database

The following procedure describes how to configure Replication server to add source database and CDC staging database.

1. Add the source database and the CDC staging database to interface file in Replication Server host.
2. Run rs_init to add the source database and the CDC staging database.

The source database and the CDC staging database are added.

2.19.4.4.3 Configuring Data Services CDC staging database in SAP Sybase ASE Server

The following procedure describes how to configure Data Services CDC staging database in SAP Sybase ASE server.

1. Select the Data Services staging database's Physical Data Model in Power Designer's Data Movement Model and right-click the context to select *Generate Script*.
2. Select *SAP Data Services CDC 14.2.0* extension in the dialog box. It generates script file suffixed with "_DataServices_CDC_DB_Runtime_Script.sql". This script file creates Data Services CDC staging database runtime environment and the provides the following functionalities:
   a) Validates the staging database to find out if this database is added to the replication server by replication server's rs_init utility.
   b) Validates if Data Services CDC staging database create script is run.
   c) Creates the Replication Server maintenance user that was defined in Data Movement Model, if it is not created before. You can change the password after creation to protect un-authorized access.
   d) Adds the maintenance user to database.
   e) Adds the TargetTable name and the CDC staging table name to DS_CDC_TABLES_MAP table for each replication definition.
   f) Adds the CDC staging table name and the retention period to DS_CHANGE_RETENTION table for each replication definition.
   g) Grants select, insert, update and delete privileges to the CDC staging tables.
   h) Grants execute privilege to "PURGE_OLD_CHANGE_DATA" stored procedure.
   i) Grants the Replication Server maintenance user, privileges to execute "Purge Old Change Data" scheduled job in SAP Sybase ASE Job Scheduler database.
   j) Creates "Purge Old Change Data" scheduled job in SAP Sybase ASE Job Scheduler database. "Purge Old Change Data" job name is SAPDS_<CDC Staging Database Name>_PURGE_CHANGE_OLD_DATA.
3. Deploy this DDL script using isql tool.

> i Note
>
> Database user who logs in must be a super user.
>
> Enter the code:
>
> ```
> isql -Usa -Ppwd -Spds -Dpdb -i DS_CDC_staging_Runtime script.sql -o
> DS_CDC_staging_Runtime_script.out
> ```
>
> where:
>
> - sa is the System Administrator user login on the CDC Staging database server.
> - pwd is the password for the System Adminstrator user login.
> - pds is the name of the DS CDC Staging database server.
> - pdb is the name of the DS CDC staging database.

2.19.4.4.4 Deploying Replication definitions and subscriptions into the Replication Server

The following procedure describes how to deploy replication definitions and subscriptions into replication server.

1. Select the Replication Process in Power Designer's Data Movement Model and right-click the context to select *Generate Script*.
2. Set *Create Connection* value to false in the dialog box and click *OK* to generate the Replication definitions, CDC Function Strings definitions subscriptions, and the Replication Server's RCL script file.

   > **Note**
   >
   > It is recommended that you create a database connection using rs_init tool.

3. Deploy this RCL script using the isql tool into the Replication Server that you configured for the CDC staging database and the source databases.

   > **Note**
   >
   > The Replication Server user who logs in to deploy must be a super user.
   >
   > Enter the code:
   >
   > `isql -Usa -Ppwd -Spds -Dpbd -i RCL_script.sql -o RCL_script.out` where:
   >
   > ○ sa is the System Administrator user login on the Replication server.
   > ○ pwd is the password for the System Administrator user login.
   > ○ pds is the name of the Replication server.
   > ○ pdb is the name of the Replication database.

2.19.4.4.5 Configuring Primary Database environment

The following procedure describes how to configure the primary database environment.

1. Select the primary database in Power Designer's Data Movement Model and right-click the context to select *Generate Script* and select *Replication Server 15.7*. It generates three script files.
2. Select the script file that contains sp_setreptable stored procedure command.
3. Check the script to make sure that “owner_on” parameter is set for those articles whose primary table name is qualified by owner.

   > **Note**
   >
   > *SAP Data Services CDC 14.2.0* extension turns on multiple-owner attribute for all those primary tables that are qualified by owner.

4. Deploy these scripts using the isql tool and the appropriate source database tool to create user, grants and permissions.

> ℹ Note
>
> The database user who deploys the script must be a super user.
>
> Enter the code:
>
> ```
> isql –Usa -Ppwd -Spdb -Dpdb -i Primary_database_Runtime script.sql -o
> Primary_database_Runtime script.out
> ```
>
> where:
>
> - sa is the System Adminstrator user login on the Primary database server.
> - pwd is the password for the System Adminstrator user login.
> - pds is the name of the Primary database server.
> - pdb is the name of the Primary database.

This marks primary tables as replicable and also change data is captured by SAP Sybase Replication Agent and propagated to Replication Server.

2.19.4.5 Configuring Data Services

To use Data Services to read and load changed data from SAP Sybase Replication Server, do the following procedures in the Designer:

- Create a CDC datastore.
- Import the metadata for SAP Sybase Replication Server tables.
- Create and define a data flow.
- Configure a CDC source.

2.19.4.5.1 Creating a CDC datastore

To access the CDC tables, create a CDC datastore using the Designer. A CDC datastore is a read-only datastore that can only access tables. Like other datastores, you can create, edit, and access a CDC datastore from the *Datastores* tab of the object library.

This procedure describes how to create a CDC datastore using SAP Sybase Replication Server.

1. Open the datastore editor.
2. Enter the name SybCDC for the datastore.
3. In the *Datastore type* box, select *Sybase RepServer CDC.*
4. In the *Database type* box, select *Sybase ASE*.
5. For *Database version*, select *Sybase ASE 15.x*.
6. Enter a *Database server name*.
7. Enter a *Database name*.
8. Enter a database *User name* and *Password*.
9. To create more than one configuration for this datastore, click *Apply* , then click *Edit* and enter the configuration details.

ℹ Note

You cannot change the database type, version, or CDC method.

10. Click *OK*.

You can now use the new datastore connection to import CDC tables in the current repository.

Related Information

Defining a database datastore [page 213]

2.19.4.5.2 Importing CDC tables

After you create a CDC datastore, you can use it to import CDC table metadata.

In the object library, right-click the datastore name and select either *Open*, *Import by Name*, or *Search*

Only the CDC tables that you selected when you set up SAP Sybase replication server for CDC are visible when you browse external metadata. You must create a CDC table in SAP Sybase ASE for every table you want to read from before you can browse and import that CDC table using SAP BusinessObjects Data Services.

Related Information

Configuring the distribution database [page 778]

2.19.4.5.3 Creating and defining the data flow

To use a SAP Sybase CDC source, you use a Query transform to remove the Sybase control columns and the Map_CDC_Operation transform to interpret the control columns and take appropriate actions:

1. In the *Designer Object Library* pane, drag the SAP Sybase CDC table, Query, and Map_CDC_Operation transforms to the data flow workspace area.

ℹ Note

- Multiple Replication Server CDC tables is allowed in same dataflow.
- The Replication Server CDC table stores data records extracted from the source database log. However, these stored data records are not equivalent to the records available in the corresponding source table. Additionally, captured change data records older than the retention period are also deleted from the Replication Server CDC table. Therefore, joining two or more such CDC tables may not produce the same result as joining their corresponding source database tables produces.

2. Configure the CDC table.
3. Add the appropriate target table and connect the objects.
4. In the *Project Area* pane, double-click a transform. The *Query Editor* opens, displaying the transform.
5. Map the Data Services control columns and the source table columns that you want in your target

The Map_CDC_Operation transform uses the values in the column in the *Row Operation Column* box to perform the appropriate operation on the source row for the target table. For the SAP Sybase CDC source table, the DI_OPERATION_TYPE column is automatically selected as the Row operation column.

The operations can be INSERT, DELETE, or UPDATE. For example, if the operation is DELETE, the corresponding row is deleted from the target table.

2.19.4.5.4 Configuring CDC source table

This procedure describes how to configure a CDC source table. Data Services applies check-points across all the tables (in a given datastore configuration) for all the data flows in a job, to provide data consistency. For more information, see the *Reference Guide*.

1. Drag a CDC datastore table into a data flow.
2. Click the name of this source object to open its source table editor.
3. Click the CDC options tab.
4. Optionally select *Enable check-point.*

Once a check-point is placed, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last check-point.

If check-points are not enabled, the CDC job reads all the rows in the CDC data source (with increased processing time.)

Related Information

Using mainframe check-points [page 758]

2.19.5 Using CDC with Microsoft SQL Server databases

SAP Data Services can capture changed data on Microsoft SQL Server databases and apply it to target systems using the following methods:

- Changed-Data Capture
- Change Tracking
- Replication Server

All three methods use the concept of check-points to read the most recent set of changes and mark the end of the read. If check-points are not enabled, Data Services reads all the changes. In addition, if you enable check-points and you run your CDC job in recovery mode, the recovered job begins to review the CDC data source at the last check-point.

To capture changed data:

- Enable CDC on the Microsoft SQL Server database
- Enable CDC on the source tables
- Configure Data Services datastores, jobs, and sources

Refer to your Microsoft documentation for details on all methods.The following table compares these methods.

Feature	Changed-Data Capture	Change Tracking	Replication Server
Microsoft SQL Server version supported	2008 and later	2008 and later	2000, 2005, 2008
Synchronous (immediate; asynchronous has latency)	No; changes are captured by the SQL Server CDC capture job.	Yes, tracks changes in line with INSERT, UPDATE, and DELETE operations so changes are available immediately.	No; changes are available after the operations are transferred by the log reader agent into the distribution database.
Requires SQL agent	Yes, a capture job populates the CDC tables.	No, the database engine populates the primary keys into change tables during DML operations.	Yes, a replication log reader agent handles the replication.
Automatic cleanup process	Yes, periodically	Yes, periodically	Yes, periodically
Requires separate tables to store tracking data	Yes, stored in a capture table. The storage depends on the number of columns captured for CDC.	Yes. Uses one internal change tracking table per source table. Uses one transaction table per database. The storage depends on number of primary key columns of the source table.	Yes, stored in a distribution database.
Historical data available	Yes	No	Yes
Requires primary key	No	Yes	Yes
Before image available for UPDATE operation in Data Services	Yes. Data Services automatically reads the before-image and makes it available for further use in the data flow.	No	Yes (optional)
Recommendation	Use when historical data or consistency across multiple tables is required.	Use for rapid synchronization.	Supported but deprecated.

2.19.5.1 Limitations

The following limitations apply to all changed-data capture methods in Microsoft SQL Server:

- You cannot use the following transforms and functions with a source table imported with a CDC datastore because of the existence of the Data Services-generated columns for CDC tables. The software cannot compare or search these columns.
  - Table_Comparison, Key_Generation, SQL, and Data_Transfer transforms. The History_Preserving transform is not supported with the Change Tracking method.
  - All database functions including lookup, lookup_ext, lookup_seq, search_replace, pushdown_sql, truncate_table, key_generation, sql, and total_rows.
- You can only create one CDC source table in a data flow.
- CDC tables are not permitted in real-time jobs.
- Profiling is not available for CDC tables.
- Displaying optimized SQL is not available for data flows that read CDC tables.
- Exporting or importing jobs with CDC tables does not retain the check-point information saved in the Data Services repository.

For the Change Tracking and Replication Server methods, do not use the same subscription name to read the same CDC table in parallel data flows because the check-point is identified by a combination of the subscription name and table name. In other words, to avoid data corruption problems, do not reuse data flows that use CDC datastores, because each time a source table extracts data, it uses the same subscription name. This means that identical jobs, depending upon when they run, can get different results and leave check-points in different locations in the file.

For the Change Tracking method, because Microsoft SQL Server cannot track the TRUNCATE operation on a source table, Data Services cannot propagate those results to the target.

2.19.5.2 Data Services columns

When you import metadata into a CDC datastore using any of the methods, Data Services adds the following columns to the original table:

- DI_SEQUENCE_NUMBER
- DI_OPERATION_TYPE

2.19.5.2.1 DI_SEQUENCE_NUMBER column

The DI_SEQUENCE_NUMBER column starts with zero at the beginning of each extraction. This field increments by one each time the software reads a row except when it encounters a pair of before- and after-images. Both the before- and after-images receive the same sequence number. This sequencing column provides a way to collate image pairs if they become separated as a result of the data flow design.

If during the course of a data flow the before- and after-images become separated or get multiplied into many rows (for example, using GROUP BY or ORDER BY clauses in a query), you could lose row order.

The Map_CDC_Operation transform allows you to restore the original ordering of image pairs by using the DI_SEQUENCE_NUMBER column as its *Sequencing column*.

For the Replication Server method, you can configure the server to retrieve before-images of UPDATE rows before Data Services applies the UPDATE operation to the target. Note that if you do not configure the Replication Server to capture before-images in the database, only after-images are captured by default. For information about when to consider using before-images, see *Using before-images* [page 750].

Related Information

Reference Guide: Transforms, Map_CDC_Operation [page 1098]

2.19.5.2.2 DI_OPERATION_TYPE column

Data Services generates values in the DI_OPERATION_TYPE column. Valid values for this column are:

- I for INSERT
- D for DELETE
- B for before-image of an UPDATE (except for the Change Tracking method, which does not use a before-image)
- U for after-image of an UPDATE

2.19.5.3 Changed-data capture (CDC) method

Using the changed-data capture (CDC) method, Data Services applies check-points across all tables (in a given datastore configuration) for all the data flows in a job to provide data consistency. Enable CDC first in the datastore editor, then select the *Enable check-point* option in the data flow source table editor. If *Enable check-point* is not selected, Data Services retrieves all the available data at that time for that table. This CDC method is not available with versions prior to Microsoft SQL Server 2008.

2.19.5.3.1 Adding a CDC datastore

This procedure describes how to create a datastore connection to Microsoft SQL Server and enable the changed-data capture (CDC) method.

1. Open the datastore editor.
2. Enter a name for the datastore.
3. In the *Datastore type* box, select *Database*.
4. In the *Database type* box, select *Microsoft SQL Server*.
5. Check the *Enable CDC* box.

6. For *Database version*, select *Microsoft SQL Server 2008* or later.
7. In the drop-down box below the *Enable CDC* option, select the *CDC* method.
8. Enter a *Database server name*.
9. Enter a *Database name*.
10. Enter a database *User name* and *Password*.
11. To create more than one configuration for this datastore, click *Apply*, then click *Edit* and enter the configuration details. Note that you cannot change the database type, version, or CDC method.
12. Click *OK*.

You can now use the new datastore connection to import the metadata of a capture instance into the current repository.

Related Information

Defining a database datastore [page 213]

2.19.5.3.2 Importing CDC metadata

After you create a CDC datastore, you can use it to import capture instance metadata.

When CDC is enabled on a table, Microsoft SQL Server creates a capture instance. A table can have up to two capture instances with different names. In the object library, right-click the datastore name and select *Open*, *Import by Name,* or *Search*. Data Services displays the capture instance name (instead of the underlying table name). Therefore, when you import by name, the name must be the capture instance name (not the table name).

The import operation adds the following columns to the original table:

Column name	Data type	Source of column
DI_SEQUENCE_NUMBER	integer	Generated by Data Services
DI_OPERATION_TYPE	varchar(1)	Generated by Data Services
MSSQL_TRAN_SEQNO	varchar(256)	Supplied by Microsoft SQL Server
MSSQL_TRAN_TIMESTAMP	timestamp	Supplied by Microsoft SQL Server
MSSQL_COLUMN_UPDATE_MASK	varchar(258)	Supplied by Microsoft SQL Server

2.19.5.3.3 Configuring a source table using CDC

This procedure describes how to configure a CDC source table when employing the Changed Data Capture (CDC) method. For more information, see the *Reference Guide*.

1. Drag a CDC datastore table into a data flow.

   The table automatically becomes a source object.

2. Click the name of this source object to open its source table editor.
3. Click the *CDC Options* tab.
4. Optionally select *Enable check-point*.

   Once a check-point is placed, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last check-point.
5. Optionally select *Automatically delete rows after reading*.

Related Information

Reference Guide: Objects, CWC table source [page 949]

2.19.5.3.4 Using CDC for data flows in a WHILE loop

For the CDC method, check-points apply at the job level. If you have data flows that run in a WHILE loop to retrieve the changes, then use the function set_cdc_checkpoint() for each iteration of the loop. This function instructs the source reader to set check-points so that the next iteration picks up the latest changes. Call this function for all the datastores used in all the data flows of the job. For more information, see the *Reference Guide*.

```
set_cdc_checkpoint(<datastore>)
```

The function returns 1 if the check-point has been successfully set; otherwise it returns 0. The value **`<datastore>`** is the name of the datastore containing the CDC tables.

Example

set_cdc_checkpoint('MyCdcSource');

2.19.5.4 Change Tracking method

The Change Tracking method identifies that rows in a table have changed but ignores how many times the row has changed or the values of any intermediate changes. Data Services retrieves only the latest data available. Therefore, change tracking is limited in the historical questions it can answer compared to the Changed-Data Capture (CDC) method. However, there is far less storage overhead because the changed data is not being captured. In addition, a synchronous tracking mechanism used to track the changes has minimal overhead to operations.

Change Tracking must first be enabled for the Microsoft SQL Server database and then enabled for the tables that you want to track within that database. Change tracking information is recorded for modified rows. The values of the primary key column from the tracked table are recorded with the change information to identify the rows that have changed. To obtain the latest data for those rows, Data Services uses the primary key column values to join the source table with the tracked table. Information about the changes can include the type of operation that caused the change (INSERT, UPDATE, or DELETE) or the columns that were changed as part of an UPDATE operation, for example.

This method is not available with versions prior to Microsoft SQL Server 2008.

2.19.5.4.1 Adding a Change Tracking datastore

This procedure describes how to create a datastore connection to Microsoft SQL Server and enable the Change Tracking method.

1. Open the datastore editor.
2. Enter a name for the datastore.
3. In the *Datastore type* box, select *Database*.
4. In the *Database type* box, select *Microsoft SQL Server*.
5. Check the *Enable CDC* box.
6. For *Database version*, select *Microsoft SQL Server 2008* or later.
7. In the drop-down box below the *Enable CDC* option, select the *Change tracking* method.
8. Enter a *Database server name*.
9. Enter a *Database name*.
10. Enter a database *User name* and *Password*.
11. To create more than one configuration for this datastore, click *Apply*, then click *Edit* and enter the configuration details. Note that you cannot change the database type, version, or CDC method.
12. Click *OK*.

You can now use the new datastore connection to import table metadata into the current repository.

Related Information

Defining a database datastore [page 213]

2.19.5.4.2 Importing Change Tracking metadata

After you create a Change Tracking datastore, you can use it to import table metadata.

In the object library, right-click the datastore name and select *Open*, *Import by Name*, or *Search*. Only the CDC tables that you select when you set up Microsoft SQL Server for Change Tracking are visible when you browse external metadata.

The import operation adds the following columns to the original table:

Column name	Data type	Source of column
DI_SEQUENCE_NUMBER	integer	Generated by Data Services
DI_OPERATION_TYPE	varchar(1)	Generated by Data Services
MSSQL_SYS_CHANGE_VERSION	decimal(19,0)	Supplied by Microsoft SQL Server

Column name	Data type	Source of column
MSSQL_SYS_CHANGE_CREATION_VERSION	decimal(19,0)	Supplied by Microsoft SQL Server
MSSQL_SYS_CHANGE_CONTEXT _MASK	varchar(256)	Supplied by Microsoft SQL Server

2.19.5.4.3 Configuring a source table to use Change Tracking

This procedure describes how to configure a CDC source table to enable Change Tracking.

1. Drag a CDC datastore table into a data flow.

   The table automatically becomes a source object.
2. Click the name of this source object to open its source table editor.
3. Click the *CDC Options* tab.
4. Specify a value for the *CDC subscription name*.

   Data Services uses this name to track the last check-point internally in the repository.
5. Select *Enable check-point*.

   Once a check-point is placed, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last check-point.

Related Information

Reference Guide: Objects, CDC table source [page 949]

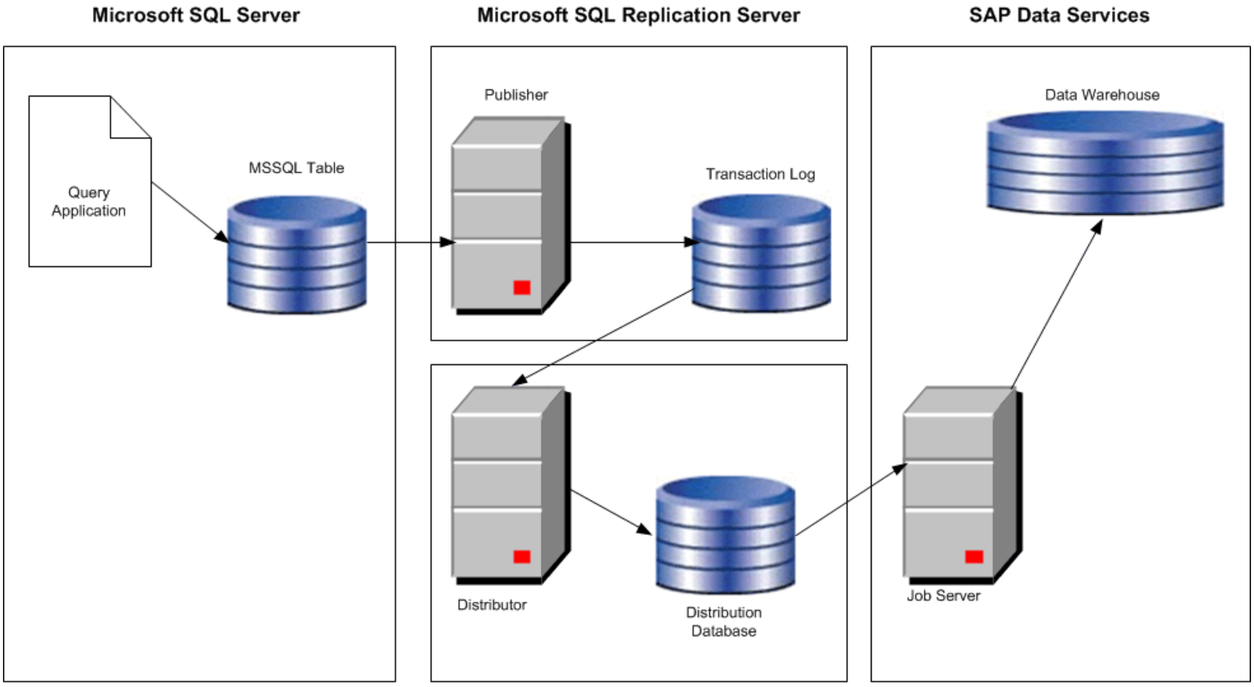
2.19.5.5 Replication Server method

Microsoft uses the following terms for the Microsoft SQL Replication Server:

- Article—An article is a table, a partition, or a database object that the DBA specifies for replication. An article can be any of the following:
  - An entire table
  - Certain columns (using a vertical filter)
  - Certain rows (using a horizontal filter)
  - A stored procedure or view definition
  - The execution of a stored procedure
  - A view
  - An indexed view
  - A user-defined function
- Distributor—The Distributor is a server that stores metadata, history data, and transactions into the distribution database. The software reads the distribution database to obtain changed data.
- Publication—A publication is a collection of one or more articles from one database. A publication makes it easier to specify a logically related set of data and database objects that you want to replicate together.

- Publisher—The Publisher is a server that makes data available for replication to other servers.
- Subscriber—A Subscriber is a server that receives replicated data. Subscribers subscribe to publications, not to individual articles within a publication. They subscribe only to the publications that they need, not to all of the publications available on a Publisher.

The software obtains changed data from the Distribution database in the Microsoft SQL Replication Server. The following diagram shows how the changed data flows from the Replication Server to Data Services.



- An application makes changes to a database and the Publisher within the Replication Server captures these changes within a transaction log.
- The Log Reader Agent in the Distributor reads the Publisher's transaction log and saves the changed data in the Distribution database.
- The software reads the data from the command table within the Distribution database, applies appropriate filters, and creates input rows for a target data warehouse table.

The software accesses the following tables within the Distribution database:

- MSarticles—contains one row for each article that a Publisher replicates.
- MSpublications—contains one row for each publication that a Publisher replicates.
- MSpublisher_databases—contains one row for each Publisher and Publisher database pair that the local Distributor services.
- MSrepl_commands—contains rows of replicated commands (changes to data).

When you enable a database for replication, Replication Server creates tables on the source database. One of these tables is Sysarticles which contains a row for each article defined in this specific database. One of the columns in Sysarticles indicates which columns in a source table are being published.

2.19.5.5.1 Configuring the distribution database

If the software connects to a Microsoft SQL Server to extract data, you need to configure the Distribution database in the Replication Server to capture changes to these tables.

2.19.5.5.1.1 Microsoft SQL Server 2000

The following steps summarize the procedure to configure the Replication Server for Microsoft SQL Server 2000 databases.

1. On the Replication node of the Microsoft SQL Enterprise Manager, select the *Configure publishing, subscribers, and the Distribution* option. Follow the wizard to create the Distributor and Distribution database.

   The following steps summarize the procedure to configure SQL Replication Server for your Microsoft SQL Server 2000 database.

   The wizard generates the following components that you need to specify on the Datastore Editor when you define a Microsoft SQL Server CDC datastore:
   - MSSQL distribution server name
   - MSSQL distribution database name
   - MSSQL distribution user name
   - MSSQL distribution password
2. Select the *New Publications* option on the Replication node of the Microsoft SQL Enterprise Manager to create new publications that specify the tables that you want to publish. The software requires the following settings in the Advanced Options:
   - Select *Transactional publication* on the Select Publication Type window. This type updates data at the Publisher and send changes incrementally to the Subscriber.
   - In the Commands tab of the Table Article Properties window:
     - If you want before images for UPDATE and DELETE commands, select XCALL. Otherwise, select CALL.
     - Clear the options *Create the stored procedures during initial synchronization of subscriptions* and *Send parameters in binary format* options because the software does not use store procedures and has its own internal format.
   - On the Snapshot tab of the Table Article Properties window:
     - Select *Keep the existing table unchanged* because the software treats the table as a log.
     - Clear *Clustered indexes* because the software treats the table as a log and reads sequentially from it.
   - Specify a publication name and description. You specify this publication name on the Datastore Editor when you define an MSSQL CDC datastore.
   - Select option *Yes, allow anonymous subscriptions* to save all transactions in the Distribution database.

For more information, see the Microsoft SQL Enterprise Manager online help.

2.19.5.5.1.2 Microsoft SQL Server 2005 and 2008

The following procedure summarizes how to configure publications for Microsoft SQL Server 2005 and 2008 databases for CDC.

1. Start the Microsoft SQL Server Management Studio.
2. Log in and go to the *Replication* node in the Object Explorer.
3. If this is the first time you are configuring distribution for this server, right-click the *Replication* node and select *Configure Distribution* in the context menu. You can configure the distribution server, snapshot folder, distribution database, and the users for this distributor.
4. Right-click the *Replication* node again and select ▶ *New* ▶ *Publication* ◀. The New Publication Wizard opens.
5. In the New Publication Wizard, click *Next.*
6. Select the database that you want to publish and click *Next.*
7. Under Publication type, select *Transactional publication*, then click *Next*.
8. Click to select tables and columns to publish as articles. Then open *Article Properties*. For each selected table, click *Set Properties of Highlighted Table Article*. The Article Properties window opens:
   a) Set the following to False: *Copy clustered index*; *Copy INSERT, UPDATE and DELETE stored procedures*; *Create schemas at subscriber*.
   b) Set the *Action if name is in use* to *Keep the existing table unchanged*.
   c) Set *Update delivery format* and *Delete delivery format* to *XCALL <stored procedure>*.
   d) Click *OK* to save the article properties.
9. Click *Next*. You can click *Add* to add row filters. Click *Next* if you do not need to filter the data in your publication.
10. Configure Agent Security and specify the account connection setting.
    a) For the Snapshot Agent, click *Security Settings*. Specify the account under which the snapshot agent will run. Configure the account with system administration privileges. Specify the account that connects to the publisher and click *OK.*
    b) For the Log Reader Agent, the *Use the security settings from the Snapshot Agent* option is selected by default. To use different settings, clear this option and click *Security Settings*. Note that it requires a login that grants system administration privileges.
11. In the Wizard Actions window, select *Create the publication* then click *Next*.
12. To complete the wizard, enter a Publication name and click *Finish*.

For more information, see the Microsoft SQL Enterprise Manager documentation.

2.19.5.5.2 Configuring Data Services

To use Data Services to read and load changed data from SQL Server databases using the Replication Server, do the following procedures in the Designer:

- Create a CDC datastore
- Import metadata for Microsoft SQL Server tables
- Configure a CDC source

2.19.5.5.2.1 Adding the CDC datastore

This procedure describes how to create a CDC datastore using the Replication method.

1. Open the Datastore Editor.
2. Enter a name for the datastore.
3. In the *Datastore type* box, select *Database*.
4. In the *Database type* box, select *Microsoft SQL Server*.
5. Check the *Enable CDC* box to enable the CDC feature.
6. Select a *Database version*.
7. Enter a *Database name* (use the name of the Replication server).
8. Enter a database *User name* and *Password*.
9. In the CDC section, enter the names that you created for this datastore when you configured the Distributor and Publisher in the Replication Server:
    - MSSQL distribution server name
    - MSSQL distribution database name
    - MSSQL publication name
    - MSSQL distribution user name
    - MSSQL distribution password
10. If you want to create more than one configuration for this datastore, click *Apply*, then click *Edit* and follow step 9 again for any additional configurations.
11. Click *OK*.

    You can now use the new datastore connection to import table metadata into the current repository.

Related Information

Defining a database datastore [page 213]

2.19.5.5.2.2 Importing CDC metadata

After you create a CDC datastore, you can use it to import CDC table metadata. In the object library, right-click the datastore name and select *Open*, *Import by Name,* or *Search*. Only the CDC tables that you select when you set up Microsoft SQL Server for CDC are visible when you browse external metadata. Data Services uses the MSpublications and MSarticles table in the Distribution database of SQL Replication Server to create a list of published tables.

When you import each CDC table, the software uses the Sysarticles table in the Publisher database of SQL Replication Server to display only published columns.

The import operation adds the following columns to the original table:

Column name	Data type	Source of column
DI_SEQUENCE_NUMBER	integer	Generated by Data Services
DI_OPERATION_TYPE	varchar(1)	Generated by Data Services
MSSQL_TRAN_SEQNO	varchar(256)	Supplied by the Replication Server
MSSQL_TRAN_TIMESTAMP	timestamp	Supplied by the Replication Server

Related Information

Configuring the distribution database [page 778]

2.19.5.5.2.3 Configuring a source table using replication

This procedure describes how to configure a CDC source table using the replication method. For more information, see the *Reference Guide*.

1. Drag a CDC datastore table into a data flow.
   The table automatically becomes a source object.
2. Click the name of this source object to open its source table editor.
3. Click the *CDC Options* tab.
4. Specify a value for the *CDC subscription name*.
   Data Services uses this name to track the last check-point internally in the repository.

Related Information

Using mainframe check-points [page 758]

2.19.5.5.2.3.1 Using check-points with replication servers

A Log Reader Agent in the Microsoft SQL Replication Server reads the transaction log of the Publisher and saves the changed data into the Distribution database, which Data Services uses as the CDC data source. Rows of changed data append to the previous load in the CDC data source.

When you enable check-points, a CDC job uses the subscription name to read the most recent set of appended rows and to mark the end of the read. If check-points are not enabled, the CDC job reads all the rows in the CDC data source (with increased processing time).

To use check-points, on the source table editor enter the *CDC Subscription name* and select the *Enable check-point* option.

2.19.5.5.2.3.2 Using before-images from Microsoft SQL Server sources

When you must capture before-image UPDATE rows:

1. Make sure the Replication Server is set up to retrieve full before-images.
2. When you create a Publication in the Replication Server, specify XCALL for UPDATE commands and DELETE commands to obtain before-images.
3. Select the *Get before-images for each update row* option in the CDC table's source editor.

After you check the *Get before-images for each update row* option, the software processes two rows for every update. In addition to the performance impact of this data volume increase, the before- and after-image pairs could be separated or lost depending on the design of your data flow, which would cause data integrity issues.

> **Note**
>
> The Map_CDC_Operation transform can resolve problems, but undesirable results can still occur due to programming errors. When you use functions and transforms that re-order, re-direct, eliminate, and multiply the number of rows in a data flow, be aware of the possible impact to targets.

Related Information

Using before-images [page 750]

Reference Guide: Transforms [page 1067]

2.19.5.5.2.3.3 Configuring CDC source table

This procedure describes how to configure a CDC source table. Data Services applies check-points across all the tables (in a given datastore configuration) for all the data flows in a job, to provide data consistency. For more information, see the *Reference Guide*.

1. Drag a CDC datastore table into a data flow.
2. Click the name of this source object to open its source table editor.
3. Click the CDC options tab.
4. Optionally select *Enable check-point*.

Once a check-point is placed, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last check-point.

If check-points are not enabled, the CDC job reads all the rows in the CDC data source (with increased processing time.)

Related Information

Using mainframe check-points [page 758]

2.19.6 Using CDC with timestamp-based sources

Use Timestamp-based CDC to track changes:

- If you are using sources other than Oracle 9i, DB2 8.2, mainframes accessed through IBM II Classic Federation, or mainframes accessed through Attunity and
- If the following conditions are true:
  - There are date and time fields in the tables being updated
  - You are updating a large table that has a small percentage of changes between extracts and an index on the date and time fields
  - You are not concerned about capturing intermediate results of each transaction between extracts (for example, if a customer changes regions twice in the same day).

It is not recommended that you use the Timestamp-based CDC when:

- You have a large table, a large percentage of it changes between extracts, and there is no index on the timestamps.
- You need to capture physical row deletes.
- You need to capture multiple events occurring on the same row between extracts.

The term timestamp refers to date, time, or datetime values. The discussion in this section applies to cases where the source table has either `CREATE` or `UPDATE` timestamps for each row.

Timestamps can indicate whether a row was created or updated. Some tables have both create and update timestamps; some tables have just one. This section assumes that tables contain at least an update timestamp.

Some systems have timestamps with dates and times, some with just the dates, and some with monotonically generated increasing numbers. You can treat dates and generated numbers the same.

It is important to note that for the timestamps based on real time, time zones can become important. If you keep track of timestamps using the nomenclature of the source system (that is, using the source time or source-generated number), you can treat both temporal (specific time) and logical (time relative to another time or event) timestamps the same way.

Related Information

Processing timestamps [page 784]
Overlaps [page 785]
Types of timestamps [page 790]

2.19.6.1 Processing timestamps

The basic technique for using timestamps to determine changes and to save the highest timestamp loaded in a given job and start the next job with that timestamp.

To do this, create a status table that tracks the timestamps of rows loaded in a job. At the end of a job, `UPDATE` this table with the latest loaded timestamp. The next job then reads the timestamp from the status table and selects only the rows in the source for which the timestamp is later than the status table timestamp.

The following example illustrates the technique. Assume that the last load occurred at 2:00 PM on January 1, 1998. At that time, the source table had only one row (key=1) with a timestamp earlier than the previous load. SAP Data Services loads this row into the target table and updates the status table with the highest timestamp loaded: 1:10 PM on January 1, 1998. After 2:00 PM the software adds more rows to the source table:

Source table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

Target table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM

Status table

Last_Timestamp
01/01/98 01:10 PM

At 3:00 PM on January 1, 1998, the job runs again. This time the job does the following:

1. Reads the Last_Timestamp field from the status table (`01/01/98 01:10 PM`).
2. Selects rows from the source table whose timestamps are later than the value of Last_Timestamp. The SQL command to select these rows is:

   ```
   SELECT * FROM Source
   WHERE 'Update_Timestamp' > '01/01/98 01:10 pm'
   ```

   This operation returns the second and third rows (key=2 and key=3).
3. Loads these new rows into the target table.
4. Updates the status table with the latest timestamp in the target table (`01/01/98 02:39 PM`) with the following SQL statement:

   ```
   UPDATE STATUS SET 'Last_Timestamp' = SELECT MAX('Update_Timestamp') FROM
   target_table
   ```

The target shows the new data:

Table 10: Source table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

Table 11: Target table

Key	Data	Update_Timestamp
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM
1	Alvarez	01/01/98 01:10 PM

Table 12: Status table

Last_Timestamp
01/01/98 02:39 PM

To specify these operations, a data flow requires the following objects (and assumes all the required metadata for the source and target tables has been imported):

- A data flow to extract the changed data from the source table and load it into the target table, such as Source > Query > Target.
  The query selects rows from SOURCE_TABLE to load to TARGET_TABLE.
  For example:

  ```
  SOURCE.UPDATE_TIMESTAMP > $Last_Update
  ```

  The query includes a where clause to filter rows between timestamps.
- A work flow to perform the following:
  1. Read the status table
  2. Set the value of a variable to the last timestamp
  3. Call the data flow with the variable passed to it as a parameter
  4. Update the status table with the new timestamp
- A job to execute the work flow

2.19.6.2 Overlaps

Unless source data is rigorously isolated during the extraction process (which typically is not practical), there is a window of time when changes can be lost between two extraction runs. This overlap period affects source-based changed-data capture because this kind of data capture relies on a static timestamp to determine changed data.

For example, suppose a table has 1000 rows (ordered 1 to 1000). The job starts with timestamp 3:00 and extracts each row. While the job is executing, it updates two rows (1 and 1000) with timestamps 3:01 and 3:02, respectively. The job extracts row 200 when someone updates row 1. When the job extracts row 300, it updates row 1000. When complete, the job extracts the latest timestamp (3:02) from row 1000 but misses the update to row 1.

Here is the data in the table:

Row Number	Column A
1	...
2	...
3	...
...	...
200	...
...	...
600	...
...	...
1000	...

There are three techniques for handling this situation:

- Overlap avoidance
- Overlap reconciliation
- Presampling

The following sections describe these techniques and their implementations in SAP Data Services. This section continues on the assumption that there is at least an update timestamp.

Related Information

Overlap avoidance [page 786]
Overlap reconciliation [page 787]
Presampling [page 787]
Types of timestamps [page 790]

2.19.6.2.1 Overlap avoidance

In some cases, it is possible to set up a system where there is no possibility of an overlap. You can avoid overlaps if there is a processing interval where no updates are occurring on the target system.

For example, if you can guarantee that the data extraction from the source system does not last more than one hour, you can run a job at 1:00 AM every night that selects only the data updated the previous day until midnight.

While this regular job does not give you up-to-the-minute updates, it guarantees that you never have an overlap and greatly simplifies timestamp management.

2.19.6.2.2 Overlap reconciliation

Overlap reconciliation requires a special extraction process that reapplies changes that could have occurred during the overlap period. This extraction can be executed separately from the regular extraction. For example, if the highest timestamp loaded from the previous job was 01/01/98 10:30 PM and the overlap period is one hour, overlap reconciliation reapplies the data updated between 9:30 PM and 10:30 PM on January 1, 1998.

The overlap period is usually equal to the maximum possible extraction time. If it can take up to **`<n>`** hours to extract the data from the source system, an overlap period of **`<n>`** (or **`<n>`** plus some small increment) hours is recommended. For example, if it takes at most two hours to run the job, an overlap period of at least two hours is recommended.

There is an advantage to creating a separate overlap data flow. A "regular" data flow can assume that all the changes are new and make assumptions to simplify logic and improve performance. For example, rows flagged as INSERT are often loaded into a fact table, but rows flagged as UPDATE rarely are. Thus, the regular data flow selects the new rows from the source, generates new keys for them, and uses the database loader to add the new facts to the target database. Because the overlap data flow is likely to apply the same rows again, it cannot blindly bulk load them or it creates duplicates. Therefore, the overlap data flow must check whether the rows exist in the target and insert only the ones that are missing. This lookup affects performance; therefore, perform it for as few rows as possible.

If the data volume is sufficiently low, you can load the entire new data set using this technique of checking before loading, avoiding the need to create two different data flows.

2.19.6.2.3 Presampling

Presampling eliminates the overlap by first identifying the most recent timestamp in the system, saving it, and then extracting rows up to that timestamp.

The technique is an extension of the simple timestamp processing technique. The main difference is that the status table now contains a start and an end timestamp. The start timestamp is the latest timestamp extracted by the previous job; the end timestamp is the timestamp selected by the current job.

To return to the example: The last extraction job loaded data from the source table to the target table and updated the status table with the latest timestamp loaded:

Source table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

Target table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM

Status table

Start_Timestamp	End_Timestamp
01/01/98 01:10 PM	NULL

Now it's 3:00 PM on January 1, 1998, and the next job runs; it does the following:

1. Selects the most recent timestamp from the source table and inserts it into the status table as the End Timestamp.
   The SQL command to select one row is:

   ```
   SELECT MAX(Update_Timestamp) FROM source table
   ```

   The status table becomes:

Status table

Start_Timestamp	End_Timestamp
01/01/98 01:10 PM	0 1/01/98 02:39 PM

1. Selects rows from the source table whose timestamps are greater than the start timestamp but less than or equal to the end timestamp. The SQL command to select these rows is:

   ```
   SELECT *
   FROM source table
   WHERE Update_Timestamp > '1/1/98 1:10pm'
   AND Update_Timestamp <= '1/1/98 2:39pm'
   ```

   This operation returns the second and third rows (key=2 and key=3)
2. Loads these new rows into the target table.
3. Updates the status table by setting the start timestamp to the previous end timestamp and setting the end timestamp to NULL.

The table values end up as follows:

Table 13: Source table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

Table 14: Target table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

Table 15: Status table

Start_Timestamp	End_Timestamp
01/01/98 02:39 PM	NULL

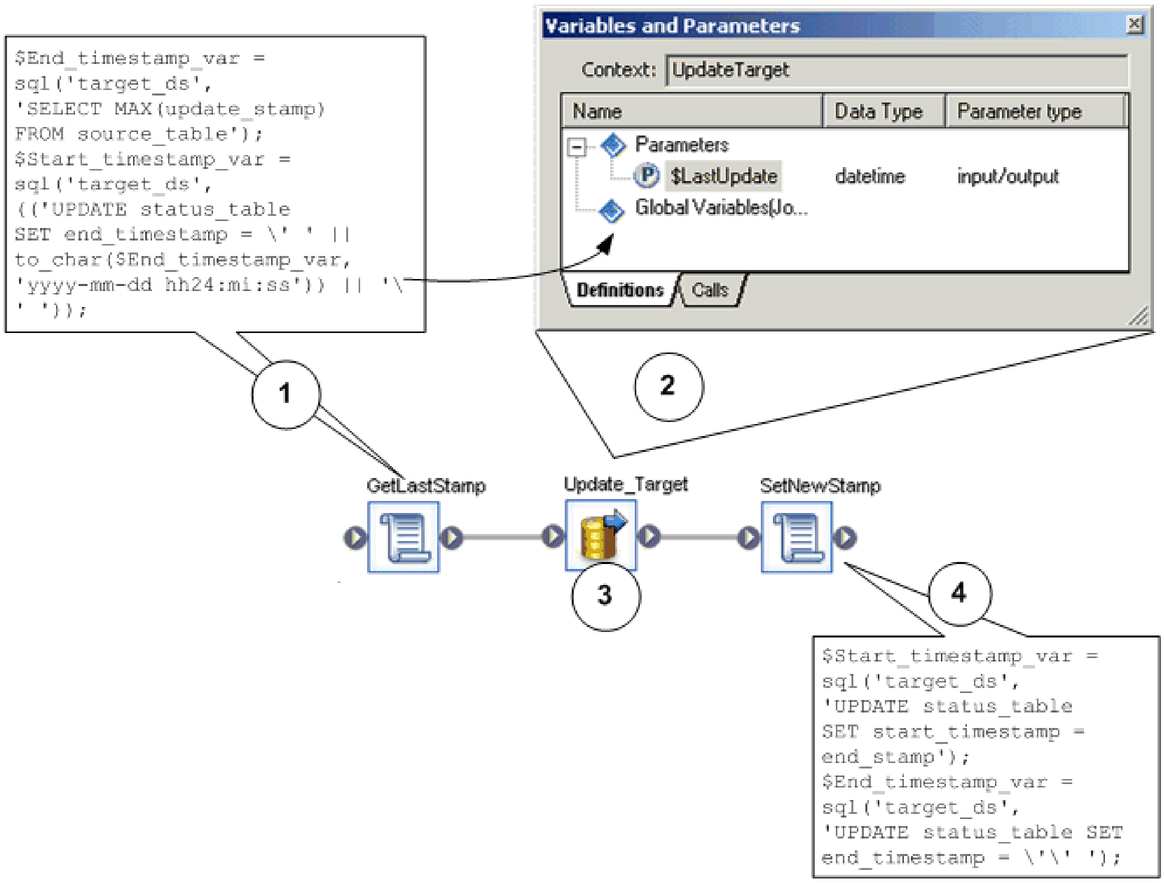
To enhance the previous example to consider the overlap time requires the following changes to the work flow:

- A data flow to extract the changes since the last update and before the most recent timestamp, such as Source > Query > Target.
  The query selects rows from SOURCE_TABLE to load to TARGET_TABLE.
  For example:

  ```
  SOURCE.UPDATE_TIMESTAMP > $start_last_update and
  SOURCE.UPDATE_TIMESTAMP < $end_last_update
  ```

  The query includes a where clause to filter rows between timestamps.
- A work flow to perform the following:
  1. Read the source table to find the most recent timestamp.
  2. Set the value of two variables to the start of the overlap time and to the end of the overlap time, respectively.
  3. Call the data flow with the variables passed to it as parameters.
  4. Update the start timestamp with the value from end timestamp and set the end timestamp to NULL.

Work flow: Changed data with timestamps



```
$End_timestamp_var =
sql('target_ds',
'SELECT MAX(update_stamp)
FROM source_table');
$Start_timestamp_var =
sql('target_ds',
(('UPDATE status_table
SET end_timestamp = \' ' ||
to_char($End_timestamp_var,
'yyyy-mm-dd hh24:mi:ss')) || '\
' '));
```

Variables and Parameters

Context: UpdateTarget

Name	Data Type	Parameter type
Parameters		
$LastUpdate	datetime	input/output
Global Variables(Jo...		

Definitions | Calls

```
$Start_timestamp_var =
sql('target_ds',
'UPDATE status_table
SET start_timestamp =
end_stamp');
$End_timestamp_var =
sql('target_ds',
'UPDATE status_table SET
end_timestamp = \'\' ');
```

Related Information

Processing timestamps [page 784]

2.19.6.3 Types of timestamps

Some systems have timestamps that record only when rows are created. Others have timestamps that record only when rows are updated. (Typically, update-only systems set the update timestamp when the row is created or updated.) Finally, there are systems that keep separate timestamps that record when rows are created and when they are updated.

Related Information

Create-only timestamps [page 791]

Update-only timestamps [page 791]

Create and update timestamps [page 791]

2.19.6.3.1 Create-only timestamps

If the source system provides only create timestamps, you have these options:

- If the table is small enough, you can process the entire table to identify the changes. The section *Using CDC for targets* [page 797], describes how to identify changes.
- If the table never gets updated, you can extract only the new rows.
- If the table is large and gets updated, you can combine the following two techniques:
  - Periodically (for example, daily) extract only the new rows.
  - Less frequently (for example, weekly) extract the updated rows by processing the entire table.

2.19.6.3.2 Update-only timestamps

Using only an update timestamp helps minimize the impact on the source systems, but it makes loading the target systems more difficult. If the system provides only an update timestamp and there is no way to tell new rows from updated rows, your job has to reconcile the new data set against the existing data using the techniques described in the section *Using CDC for targets* [page 797].

2.19.6.3.3 Create and update timestamps

Both timestamps allow you to easily separate new data from updates to the existing data. The job extracts all the changed rows and then filters unneeded rows using their timestamps.

Accomplish these extractions in the software by adding the WHERE clause from the following SQL commands into an appropriate query transform:

- Find new rows:

  ```
  SELECT * FROM source_table
  WHERE Create_Timestamp > $Last_Timestamp
  ```

- Find updated rows:

  ```
  SELECT * FROM source_table
  WHERE Create_Timestamp <= $Last_Timestamp AND
  Update_Timestamp > $Last_Timestamp)
  ```

From here, the new rows go through the key-generation process and are inserted into the target, and the updated rows go through the key-lookup process and are updated in the target.

For performance reasons, you might want to separate the extraction of new rows into a separate data flow to take advantage of bulk loading into the target. The updated rows cannot be loaded by bulk into the same target at the same time.

2.19.6.4 Timestamp-based CDC examples

2.19.6.4.1 Preserving generated keys

For performance reasons, many data warehouse dimension tables use generated keys to join with the fact table. For example, customer ABC has a generated key 123 in the customer dimension table. All facts for customer ABC have 123 as the customer key. Even if the customer dimension is small, you cannot simply reload it every time a record changes: unless you assign the generated key of 123 to the customer ABC, the customer dimension table and the fact tables do not correlate.

You can preserve generated keys by either using the lookup function or comparing tables.

Related Information

Using the lookup function [page 792]

Comparing tables [page 794]

2.19.6.4.1.1 Using the lookup function

If history preservation is not an issue and the only goal is to generate the correct keys for the existing rows, the simplest technique is to look up the key for all rows using the `lookup` function in a query. If you do not find the key, generate a new one.

In the following example, the customer dimension table contains generated keys. When you run a job to update this table, the source customer rows must match the existing keys.

Source customer table

Company Name	Customer ID
ABC	001
DEF	002
GHI	003
JKL	004

Target dimension table

Gen_Key	Company Name	Customer ID
123	ABC	001
124	DEF	002

Gen_Key	Company Name	Customer ID
125	GHI	003

This example data flow does the following:

1. Extracts the source rows.
2. Retrieves the existing keys using a `lookup` function in the mapping of a new column in a query.
3. Loads the result into a file (to be able to test this stage of the data flow before adding the next steps).

The `lookup` function compares the source rows with the target. The arguments for the function are as follows:

lookup function arguments	Description
target_ds.owner.customer	Fully qualified name of the target table containing the generated keys.
GKey	The column name in the target table containing the generated keys.
NULL	NULL value to insert in the key column if no existing key is found.
'PRE_LOAD_CACHE'	Caching option to optimize the lookup performance.
Customer_ID	The column in the target table containing the value to use to match rows.
Customer_ID	The column in the source table containing the values to use to match rows.

The resulting data set contains all the rows from the source with generated keys where available:

Result data set

Gen_Key	Company Name	Customer ID
123	ABC	001
124	DEF	002
125	GHI	003
NULL	JKL	004

Adding a new generated key to the new records requires filtering out the new rows from the existing and updated rows. In the data flow, this requires the following steps: A query to select the rows with NULL generated keys. A Key_Generation transform to determine the appropriate key to add. A target to load the new rows into the customer dimension table.

This data flow handles the new rows; however, the rows from the source whose keys were found in the target table might contain updated data. Because this example assumes that preserving history is not a requirement, the software loads all rows from the source into the target.

2.19.6.4.1.1.1 Handling updated rows in the data flow

The data flow requires new steps to handle updated rows, as follows:

1. A new line leaving the query that looked up the existing keys.
2. A query to filter the rows with existing keys from the rows with no keys.
3. A target to load the rows into the customer dimension table.

2.19.6.4.1.2 Comparing tables

The drawback of the generated-keys method is that even if the row has not been changed, it generates an UPDATE and is loaded into the target. If the amount of data is large, a table-comparison transform provides a better alternative by allowing the data flow to load only changed rows.

The table-comparison transform examines all source rows and performs the following operations:

- Generates an INSERT for any new row not in the target table.
- Generates an UPDATE for any row in the target table that has changed.
- Ignores any row that is in the target table and has not changed.
- Fills in the generated key for the updated rows.

You can then run the result through the key-generation transform to assign a new key for every INSERT. This is the data set that the software loads into the target table.

The data flow that accomplishes this transformation includes the following steps:

1. A source to extract the rows from the source table(s).
2. A query to map columns from the source.
3. A table-comparison transform to generate INSERT and UPDATE rows and to fill in existing keys.
4. A key-generation transform to generate new keys.
5. A target to load the rows into the customer dimension table.

2.19.6.4.2 Preserving history

History preserving allows the data warehouse or data mart to maintain the history of data so you can analyze it over time. Most likely, you will perform history preservation on dimension tables.

For example, if a customer moves from one sales region to another, simply updating the customer record to reflect the new region would give you misleading results in an analysis by region over time because all purchases made by a customer before the move would incorrectly be attributed to the new region.

SAP Data Services provides a special transform that preserves data history to prevent this kind of situation. The History_Preserving transform ignores everything but rows flagged as UPDATE. For these rows, it compares the values of specified columns and, if the values have changed, flags the row as INSERT. This produces a second row in the target instead of overwriting the first row.

To expand on how the software would handle the example of the customer who moves between regions:

- If `Region` is a column marked for comparison, the History_Preserving transform generates a new row for that customer.
- A Key_Generation transform gives the new row a new generated key and loads the row into the customer dimension table.
- The original row describing the customer remains in the customer dimension table with a unique generated key.

In the following example, one customer moved from the East region to the West region, and another customer's phone number changed.

Source Customer table

Customer	Region	Phone
Fred's Coffee	East	(212) 123-4567
Jane's Donuts	West	(650) 222-1212
Sandy's Candy	Central	(115) 231-1233

Target Customer table

GKey	Customer	Region	Phone
1	Fred's Coffee	East	(212) 123-4567
2	Jane's Donuts	East	(201) 777-1717
3	Sandy's Candy	Central	(115) 454-8000

In this example, the data flow preserves the history for the `Region` column but does not preserve history for the `Phone` column. The data flow contains the following steps:

1. A source to extract the rows from the source table(s).
2. A query to map columns from the source.
3. A table-comparison transform to generate INSERTs and UPDATEs and to fill in existing keys.
4. A History_Preserving transform to convert certain `UPDATE` rows to INSERT rows.
5. A key-generation transform to generate new keys for the updated rows that are now flagged as INSERT.
6. A target to load the rows into the customer dimension table.

Now that there are two rows for Jane's Donuts, correlations between the dimension table and the fact table must use the highest key value.

Note that updates to non-history preserving columns update all versions of the row if the update is performed on the natural key (for example, `Customer`), and only update the latest version if the update is on the generated key (for example, `GKey`). You can control which key to use for updating by appropriately configuring the loading options in the target editor.

2.19.6.4.2.1 valid_from date and valid_to date

To support temporal queries like "What was the customer's billing address on May 24, 1998," SAP Data Services supports *Valid from* and *Valid to* date columns.

In history-preserving techniques, there are multiple records in the target table with the same source primary key values. A record from the source table is considered valid in the dimension table for all date values `t` such that the *Valid from* date is less than or equal to `t`, which is less than the *Valid to* date. (Valid in this sense means that the record's generated key value is used to load the fact table during this time interval.)

When you specify the *Valid from* and *Valid to* entries, the History_Preserving transform generates an UPDATE record before it generates an INSERT statement for history-preservation reasons (it converts an UPDATE into an INSERT). The UPDATE record will set the *Valid to* date column on the current record (the one with the same primary key as the INSERT) to the value in the *Valid from* date column in the INSERT record.

2.19.6.4.2.2 Update flag

To support slowly changing dimension techniques, Data Services enables you to set an update flag to mark the current record in a dimension table.

Value *Set value* in column *Column* identifies the current valid record in the target table for a given source table primary key.

When you specify *Column*, the History_Preserving transform generates an UPDATE record before it generates an INSERT statement.

This UPDATE record will set the *Column* value to *Reset value* in the target table record with the same source primary key as the INSERT statement.

In the INSERT statement the *Column* will be set to *Set value*.

When you specify entries in both the groups, the History_Preserving transform generates only one extra UPDATE statement for every INSERT statement it produces. This UPDATE statement updates the *Valid to* value.

2.19.6.5 Additional job design tips

When designing a job to implement changed-data capture (CDC), you must consider:

- Synchronizing header and detail
- Capturing physical deletions

2.19.6.5.1 Synchronizing header and detail

Typically, source systems keep track of header and detail information changes in an independent way. For example, if a line-item status changes, its "last modified date" column updates, but the same column at the order header level does not update. Conversely, a change to the default ship-to address in the order header might impact none of the existing line items.

In some instances, however, your source system might not consistently update those tracking columns, or you might not have access to such information (for example, when rows are physically deleted). In these cases, you might choose to extract all header and detail information whenever any changes occur at the header level or in any individual line item.

To extract all header and detail rows when any of these elements have changed, use logic similar to this SQL statement:

SELECT ... FROM HEADER, DETAIL WHERE HEADER.ID = DETAIL.ID AND (HEADER.LAST_MODIFIED BETWEEN $G_SDATE AND $G_EDATE OR DETAIL.LAST_MODIFIED BETWEEN $G_SDATE AND $G_EDATE)

For some databases, this WHERE clause is not well optimized and might cause serious performance degradation. You might opt to relax that clause by removing one of the upper bounds, such as in:

... WHERE HEADER.ID = DETAIL.ID AND (HEADER.LAST_MODIFIED BETWEEN $G_SDATE AND $G_EDATE OR DETAIL.LAST_MODIFIED >= $G_SDATE) ...

This might retrieve a few more rows than originally intended, but it might improve the final performance of your system while not altering the result of your target database.

2.19.6.5.2 Capturing physical deletions

When your source system allows rows to be physically deleted, your job should include logic to update your target database correspondingly. There are several ways to do this:

- Scan a log of operations — If your system logs transactions in a readable format or if you can alter the system to generate such a log, then you can scan that log to identify the rows you need to delete.
- Perform a full refresh — Simply reload all of the data, therefore fully synchronizing the source system and the target database.
- Perform a partial refresh based on a data-driven time-window — For example, suppose that the source system only allows physical deletion of orders that have not been closed. If the first non-closed order in your source table occurred six months ago, then by refreshing the last six months of data you are guaranteed to have achieved synchronization.
- Perform a partial refresh based on a business-driven time-window — For example, suppose that the business that the job supports usually deletes orders shortly after creating them. In this case, refreshing the last month of orders is appropriate to maintain integrity.
- Check every order that could possibly be deleted — You must verify whether any non-closed order has been deleted. To be efficient, this technique requires you to keep a record of the primary keys for every object that is a candidate for deletion.

When physical deletions of detail information in a header-detail relationship are possible (for example, removing line items from an existing order), then you must capture these physical deletions when synchronizing header and detail information.

2.19.7 Using CDC for targets

Source-based changed-data capture is almost always preferable to target-based capture for performance reasons. Some source systems, however, do not provide enough information to make use of the source-based changed-data capture techniques. Target-based changed-data capture allows you to use the technique when source-based change information is limited.

2.20 Monitoring Jobs

2.20.1 Administrator

The Administrator application in the Management Console is the primary monitoring resource for all jobs designed in the Designer. For detailed information, see the “Administrator” section of the *Management Console Guide*.

2.21 Multi-user Development

About multiple users

SAP Data Services supports a multi-user development environment. A team can work together on an application during the development, testing, or production phase. Also, different teams can work on the different phases at the same time.

Each individual developer works on an application in their unique local repository. Each team uses a central repository to store the master copy of its application. The central repository preserves all versions of all objects in the application so you can revert to a previous version if necessary.

Related Information

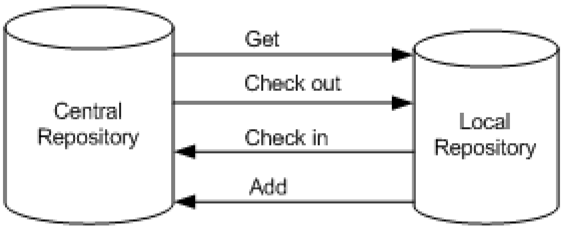
Central versus local repository [page 798]
Multiple users [page 799]

2.21.1 Central versus local repository

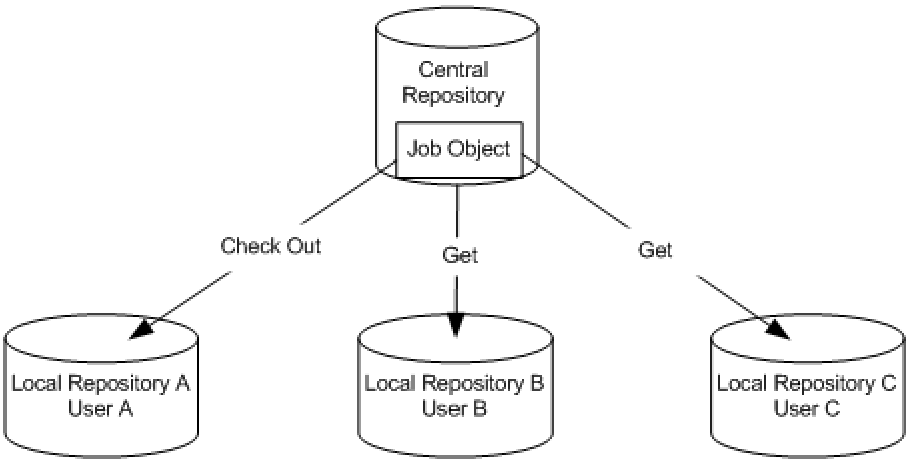
You can create a central repository for storing the team copy of a SAP Data Services application. The central repository contains all information normally found in a local repository such as definitions for each object in an application. However, the central repository is merely a storage location for this information. To change the information, you must work in a local repository.

A local repository provides a view of the central repository. You can "get" (copy) objects from the central repository into your local repository. However, to make changes to an object, you must "check out" that object from the central repository into your local repository. While you have an object checked out from the central repository, other users cannot check out that object, so they cannot change the information.

After completing changes, you "check in" the changed object. When you check in objects, the software saves the new, modified objects in the central repository.



Multiple users working from unique local repositories can connect to the same central repository. These users can work on the same application and share their work. However, at any given time only one user can check out and change a particular object. While an object is checked out to one user, other users can "get" (obtain a copy of) the object but cannot make changes that will update the central repository.


Central Repository
Job Object
Check Out
Get
Get
Local Repository A User A
Local Repository B User B
Local Repository C User C

The central repository retains history for each object. Therefore, if you find you made a change that did not work as planned, you can revert to a previous version of the object.

The local repository and the central repository must use the same software repository version. For example, you can run SAP Data Services Designer X.2 with a central and local repository version X.1. However, you cannot run SAP Data Services X.2 with a central repository X.1 and a local repository X.2

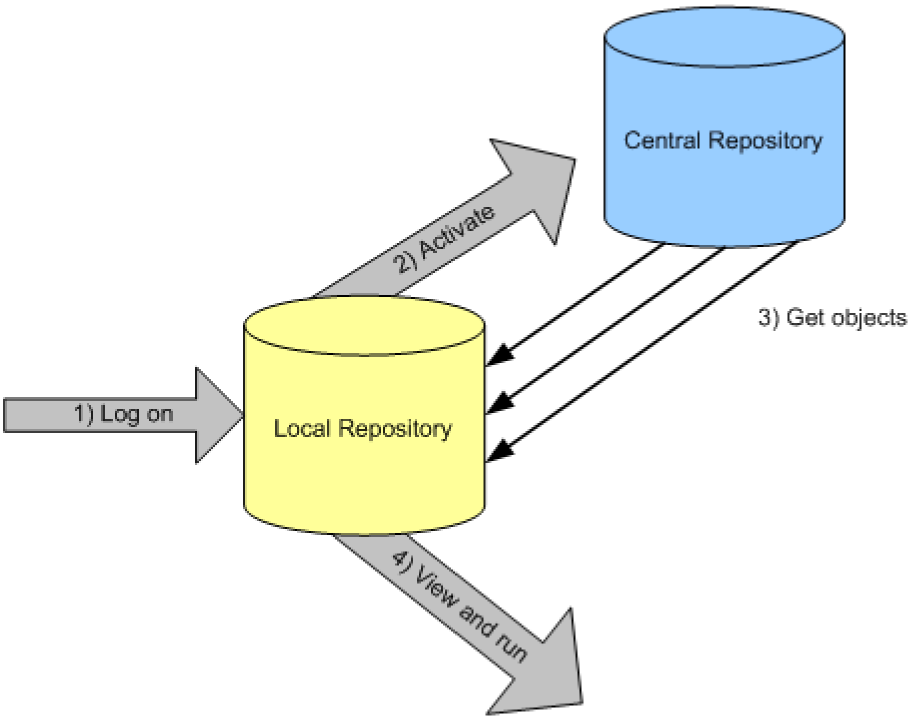
2.21.2 Multiple users

A multi-user environment affects how you use SAP Data Services and how you manage different phases of an application. For success in a multi-user environment, you must maintain consistency between your local repository and the central repository.

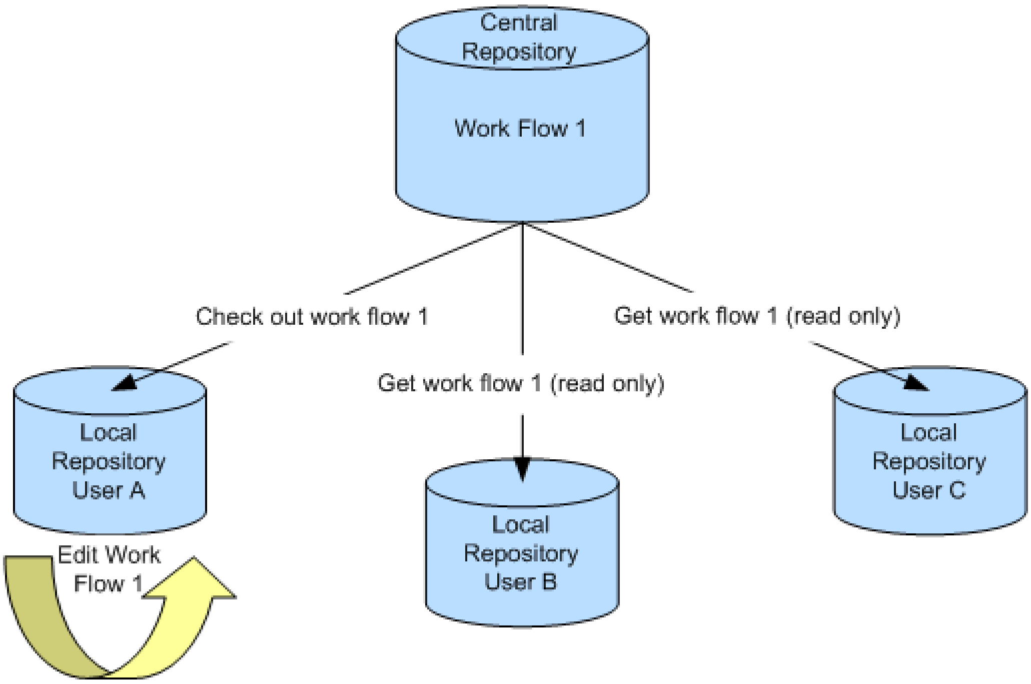
The following terms apply when discussing the software and multi-user environments:

Term	Definition
Highest level object	The highest level object is the object that is not a dependent of any object in the object hierarchy. For example, if Job 1 is comprised of Work Flow 1 and Data Flow 1, then Job 1 is the highest level object.
Object dependents	Object dependents are objects associated beneath the highest level object in the hierarchy. For example, if Job 1 is comprised of Work Flow 1 which contains Data Flow 1, then both Work Flow 1 and Data Flow 1 are dependents of Job 1. Further, Data Flow 1 is a dependent of Work Flow 1.
Object version	An object version is an instance of an object. Each time you add or check in an object to the central repository, the software creates a new version of the object. The latest version of an object is the last or most recent version created.

When working in a multi-user environment, you activate the link between your local repository and the corresponding central repository each time you log in. To ensure that your repository is current, you can get (copy) the latest version of each object in the central repository. Once you get an application in your local repository, you can view and run it from the Designer.



However, if you plan to make changes to objects in the application, you must check out those objects. After you check out an object, no other user can make changes. Essentially, you lock the version in the central repository; only you can change that version. Other users can only get and view the object.



When you are done making changes to an object, save those changes in the local repository and check the object back into the central repository. The software saves the changed object in the central repository and makes the object available for check-out by others. The software maintains all versions of saved objects in the central repository. Thus later, you can copy an old version of a saved object, even after replacing it in your local repository with a new version.

At any time, you can label an object or a group of objects. An object label provides a convenient mechanism for identifying objects later. For example, you may find it helpful to label objects by feature. Later, if you decide you want to eliminate a recently-added feature, you can get all objects that have the label without that feature.

You can also compare two objects—such as two different object versions in the central repository, or an object in your local repository to an object in the central repository. By comparing two objects, you can determine what parts of an object changed and decide whether you want to revert to an older version of an object.

Related Information

Designer Guide: Design and Debug, Comparing Objects [page 714]

2.21.3 Security and the central repository

You also have several options to make your central repository secure. Use these options when you need to control access and provide for object tracking within your central repository. These security options apply only to central repositories and include:

- Authentication — Allows only valid users to log in to a central repository.

- Authorization — Grants various levels of permissions to objects.
- Auditing — Maintains a history of changes made to an object including user names.

Implement security for a central repository by establishing a structure of groups and associated users using the Administrator.

Related Information

Implementing Central Repository Security [page 805]

2.21.4 Multi-user Environment Setup

Overview of multi-user setup

To support multiple developers, configure a multi-user environment and set up several repositories. Specifically, you must:

- Create a local repository for each developer.
- Create a central repository.
- Define a connection to central repository from each local repository.
- Activate the connection to a central repository.

Related Information

Create a nonsecure central repository [page 802]
Define a connection to a nonsecure central repository [page 803]
Activating a central repository [page 803]

2.21.4.1 Create a nonsecure central repository

To support multiple users in a single development environment, it is recommended that you use a central repository. The central repository stores master information for the development environment.

This procedure applies to nonsecure repositories only.

1. Create a database to be used for the central repository using your database management system.
2. Choose *Start* > *Programs* > *SAP Data Services 4.2* > *Data Services Repository Manager*.
3. In the Repository Manager window, click the *Central* button in the *Repository Type* field, and enter the database connection information for the central repository.

4. Click *Create*.

   The repository tables are created in the database you identified.

Related Information

Implementing Central Repository Security [page 805]

2.21.4.2 Define a connection to a nonsecure central repository

A team working on an application only needs one central repository. However, each team member requires a local repository. Furthermore, each local repository requires connection information to any central repository it must access.

This procedure applies to nonsecure repositories only.

> Note
>
> The version of the central repository must match the version of the local repository.

1. Start the Designer and log in to your local repository.
2. Choose *Tools* > *Central Repositories* to open the Options window.

   The Central Repository Connections option is selected in the Designer Options list.
3. Right-click in the *Central Repository Connections* box and select *Add*.

   The *Repository Password* window opens.
4. Enter the password for the central repository.
   The repository appears in the *Central repository connections* box.
5. Click *Activate*.
6. Again, enter the password for the central repository.
7. Click *OK*.

Related Information

Implementing Central Repository Security [page 805]

2.21.4.3 Activating a central repository

To connect to a central repository, you must activate the link between your local repository and a specific central repository.

ℹ Note

When you start the Designer, always log in to a local repository. Never log into a central repository. If you do, then the central repository acts as a local repository. Then you run the risk of corrupting version information. If you attempt to log in to the central repository, you will see a warning message. You should log out immediately and log into a local repository.

Your local repository provides a view of the objects in the active central repository. Whenever you get or check out objects, you copy objects from the active central repository. Whenever you check in objects, you save the version from your local repository into the active central repository.

You must activate the correct central repository each time you log in. When you activate a central repository, the central object library opens and shows all the objects in the central repository and the check-out status of each object.

2.21.4.3.1 To activate a central repository

1. Choose *Tools* > *Central Repositories* to open the Options window.

   The Central Repository Connections option is selected in the Designer Options list.
2. In the *Central repository connections* list, determine a central repository to make active.
3. Check *Reactivate automatically* if you want the active central repository to be reactivated when you next log on to this local repository.
4. Right-click the central repository and select *Activate*.

   The central object library opens. The Options window indicates that the selected central repository is active and closes automatically.

2.21.4.3.2 To open the central object library

Click the *Central Object Library* button on the toolbar.

The central object library looks like the object library—it shows all the objects in the repository, grouped on appropriate tabs.

The window opens in floating mode. Drag the window to dock it. To change the docking state, right-click the Central Object Library tool bar and toggle *Docking*.

You can also change central repository connection information from the central object library.

2.21.4.3.3 To change the active central repository

Select a central repository from the list on the top of the central object library.

SAP Data Services makes the selected central repository active—objects from that repository appear in the central object library. Connection information about that repository appears in the upper right corner of the central object library.

2.21.4.3.4 To change central repository connections

1. Click the *Edit Central Repository Connection* button on the top of the central object library.

   The Options window opens with the Central Repository Connections option selected in the Designer Options list.

   Alternatively, you can open the Options window by selecting *Tools* > *Central Repositories*.
2. Select a central repository in the *Central Repository Connections* box, right-click, and select *Edit*.

   When the Datastore Administrator window opens:

   - To disconnect from the currently active central repository, right-click the central repository in the *Central Repository Datastores* box and select *Deactivate*
   - To delete connection information for a central repository, right-click the central repository in the *Central Repository Datastores* box and select *Delete*.
     After confirming your selection, the connection information from this local repository is deleted. You can no longer connect to that central repository from this local repository.

     > Note
     >
     > You are not deleting the central repository; you are only deleting the connection information between your local repository and this central repository.
   - To make another repository the active central repository, right-click the central repository in the *Central Repository Datastores* box and select *Activate*.

2.21.5 Implementing Central Repository Security

About this section

This section describes how to implement optional security features for central repositories.

2.21.5.1 Overview

SAP Data Services provides options for managing secure access and tracking for objects in central repositories. Mechanisms for managing central repository security include:

- Authentication — Allows only valid users to log in to a central repository.
- Authorization — Grants various levels of permissions to objects.

- Auditing — Maintains a history of changes made to an object including user names.

Note that these security mechanisms and procedures apply only to central repositories.

2.21.5.1.1 Group-based permissions

You implement security for a central repository by establishing a structure of groups and associated users using the Administrator and the Central Management Console (CMC).

Access permissions for objects apply at the group level. More than one group can have the same permissions to the same object at a time. Groups are specific to a repository and are not visible in any other local or central repository.

Therefore, users do not get individual permissions. In the Designer, users select from the group(s) to which they belong, and the selected (current) group dictates their access to that object. Each user must have one default group but can belong to more than one group. When a user adds an object to a secure central repository, the user's current group automatically has Full permissions to that object.

User name and password authentication is required for every logon to a secure central repository. Users can change their passwords at any time in the CMC.

Related Information

Managment Console Guide: Central Repository management, Setting up users and groups [page 1881]
Administrator Guide: User management [page 46]

2.21.5.1.2 Permission levels

Each object in a secure central repository can have one of the following permissions levels:

- Full — This is the highest level of permission. The group can perform all possible actions including checking in, checking out, and deleting the object. You might assign this type of access to developers, for example.
- Read — Users can only get a copy of the object from the central repository or compare objects between their local and central object libraries. You might assign this type of access to QA, for example.
- None — Users cannot get copies of the object but can view it and its properties.

When an authenticated user adds an object to a secure central repository, the user's current group receives Full permissions to the object. All other groups receive Read permissions. Members of the group with Full permissions can change the other groups' permissions for that object.

2.21.5.1.3 Process summary

You implement security for a central repository by:

1. Using the Repository Manager to add a secure central repository or upgrade an existing nonsecure central repository.
2. Using the Central Management Console (CMC) to add users.
3. Using the Administrator to add the users to central repository groups.
4. Defining the connection from the Designer.
5. Adding objects to the central repository as well as view and modify object permissions.

Related Information

Creating a secure central repository [page 807]
Adding a multi-user administrator (optional) [page 808]
Setting up groups and users [page 808]
Defining a connection to a secure central repository [page 809]
Working with objects in a secure central repository [page 809]

2.21.5.2 Creating a secure central repository

The first step in establishing security measures for multi-user development is to create a secure central repository or upgrade an existing nonsecure central repository.

> ℹ Note
>
> These procedures apply to secure repositories only.

Related Information

Multi-user Environment Setup [page 802]

2.21.5.2.1 To create a secure central repository

1. Create a database to be used for the central repository using your database management system.
2. Choose *Start* › *Programs* › *SAP Data Services 4.2* › *Data Services Repository Manager*.
3. In the Repository Manager window, click the *Central* button in the Repository Type field and enter the database connection information for the central repository.
4. Select the *Enable security* check box.
5. Click *Create*.

   The software creates repository tables in the database you identified.

2.21.5.2.2 To upgrade a central repository from nonsecure to secure

You can modify an existing central repository to make it secure; however, you cannot undo this change.

1. Open the Repository Manager.
2. In the Repository Manager window, click the *Central* button in the Repository Type field and enter the database connection information for the central repository to modify.
3. Select the *Enable security* check box.
4. Click *Upgrade*.

   The software updates the repository tables in the database you identified.

   > Note
   >
   > When you upgrade an existing non-secure central repository to a secure central repository, a new group, DIGroup, is automatically created for you and displayed in *Management Console* > *Administrator* > *Central Repositories*. To access the repository, add existing users to the group in the Administrator.

2.21.5.3 Adding a multi-user administrator (optional)

In the Central Management Console (CMC), you have the option of adding a user with the role of Multi-user Administrator. This role is limited to managing secure central repositories, so it is therefore a subset of the Administrator role. For example, Multi-user Administrators cannot add a local repository or a nonsecure central repository.

Multi-user Administrators can:

- Add and remove secure central repositories.
- Manage users and groups.
- View secure central repository reports.

Related Information

Administrator Guide: User management [page 46]

2.21.5.4 Setting up groups and users

The next step in implementing central repository security is to add and configure groups and users with the Central Management Console (CMC) and the Administrator.

Related Information

Administrator Guide: User management [page 46]
Management Console Guide: Central Repository management, Setting up users and groups [page 1881]

2.21.5.5 Defining a connection to a secure central repository

The next step in implementing central repository security is to define a connection to the repository in the Designer.

This procedure applies to secure central repositories only.

1. Start the Designer and log in to your local repository.
2. From the *Tools* menu, click *Central Repositories* to open the Options window.
   The Central Repository Connections option should be selected in the Designer list.
3. Click *Add*.
4. Enter your CMS connection information and click *Log On*.
5. Select the secure central repository you want to connect.
6. Click *OK*.
   The list of central repository connections now includes the newly connected central repository and it is identified as being secure.
7. With the repository selected, click *Activate*.
8. Click *OK*.

Related Information

Multi-user Environment Setup [page 802]
Activating a central repository [page 803]

2.21.5.6 Working with objects in a secure central repository

Related Information

Adding objects to the central repository [page 811]
Viewing and modifying permissions [page 810]

2.21.5.6.1 Viewing and modifying permissions

After completing all configuration tasks and adding objects to the secure central repository, use the central object library to view and modify group permissions for objects.

2.21.5.6.1.1 To view permissions for an object

1. Start the Designer and log in to your local repository.
2. Open the secure central object library.

Your default group appears in the drop-down list at the top of the window and is marked with an asterisk. The Permissions column displays the current group's access level for each object. If you add a new object to the central library, the current group gets FULL permissions and all other groups get READ permission.

2.21.5.6.1.2 To change object permissions to other groups

You must have Full permissions to change object access to other groups.

1. In the central object library, right-click the object and click *Permission* > *CDC Adapter Configuration* > *Object* or *Permission* > *Object and dependants*.
2. The *Permission* dialog box opens, which displays a list of available groups and the group's access level for the object(s).
3. Click in the *Permission* column, and from the drop-down list select a permission level for the group.
4. Click *Apply* or *OK*.

2.21.5.6.1.3 To change the current group or the default group

1. To change the current group, in the central object library select a group from the drop-down box.
2. To change your default group, select the desired group from the drop-down box and click the save icon.

   The software marks the default group with an asterisk.

2.21.6 Working in a Multi-user Environment

To obtain optimal results from development in a multi-user environment, it is recommended certain processes, such as checking in and checking out objects that you change, and establishing a set of conventions that your team follows, such as labeling objects.

2.21.6.1 Filtering

SAP Data Services allows you to customize by filtering (selectively changing) environment-specific information in object definitions. Application objects can contain repository-specific information. For example, datastores and database tables might refer to a particular database connection unique to a user or a phase of development. When multiple users work on an application, they can change repository-specific information.

Specifically, filtering allows you to:

- Change datastore and database connection information
- Change the root directory for files associated with a particular file format
- Select or clear specific dependent objects

The filtering process is available when adding, checking in, checking out, or getting labeled or latest objects in a central repository.

When you select any command that uses the filtering option:

1. The *Version Control Confirmation* window displays your selected object and any dependent objects. You can exclude objects by selecting the object and changing the *Target status* from *create* or *replace* to *exclude*.
2. The *Datastore Options* window shows any datastores used by the object. This window only opens if the objects that you are adding, checking in, or checking out include a datastore.

2.21.6.2 Adding objects to the central repository

After creating a central repository, connecting it to the local repository, and activating the central repository, you can add objects from the local repository to the central repository. Remember that you do all design work—the creation of jobs, work flows, and data flows—in a local repository. Therefore, you use a local repository for the initial creation of any objects in an application. After the initial creation of an object, you add it to the central repository. Once in the central repository, the object is subject to version control and can be shared among users.

You can add a single object to the central repository, or you can add an object with all of its dependents to the central repository. When you add a single object, such as a data flow, you add only that object. No dependent objects are added.

You can add objects to the central repository at any point. However, you cannot add an object that already exists in the central repository.

You cannot add a read-only transform configuration to the repository. You can, however, replicate a transform configuration and add the replica to the repository.

2.21.6.2.1 To add a single object to the central repository

1. Open the local object library.
2. Right-click the object and select *Add to Central Repository* > *Object*.
3. The Comments window opens. Enter any comments in the *Comments* field, and click *OK*.

   The software adds the object to the active central repository.

ℹ Note

The Add to Central Repository command is not available if the object already exists in the central repository.

2.21.6.2.2 To add an object and its dependent objects to the central repository

1. Open the local object library.
2. Right-click the object and select either *Add to Central Repository* > *Object and dependents* or *Add to Central Repository* > *With filtering* (if filtering is required).
3. The Comments window opens. Enter any comments in the *Comments* field, and click *OK*.
4. If you selected *With filtering*, complete the filtering windows.
5. Click *Finish* to add the selected objects.

Alternatively, you can select the object and drag it to the central object library to add the object and its dependents to the central repository. The filtering windows are displayed.

ℹ Note

The *Add to Central Repository* command is not available if the object already exists in the central repository. However, the *Add to Central Repository* command is available if the object's dependents already exist in the central repository but the object itself does not.

ℹ Note

You cannot add a read-only transform configuration to the repository. To do so, you must create a new repository, upgrade the existing repository, or import ATL that contains a new version of read-only transform configurations.

Related Information

Filtering [page 811]

2.21.6.3 Checking out objects

When you might change any of the objects in an application, you should check out the objects that you expect to change. When you check out an object, you make that object unavailable to other users—other users can view the object but cannot make changes to the object. Checking out an object ensures that two users do not make conflicting changes to the object simultaneously.

Data Services changes the object icons in both the local and central object libraries to indicate that the object is checked out.

When an object is checked out, your central object library shows you the local repository that has checked out the object. Based on the repository name, you can determine which user is working with that object.

To see periodic changes, refresh the central object library by clicking on the *Refresh Central Object Library* button in the toolbar of the central object library.

Choose a check-out command based on what you will do to an object.

2.21.6.3.1 Check out single objects or objects with dependents

Dependents are objects used by another object—for example, data flows that are called from within a work flow. You can check out a single object or an object with all of its dependents (as calculated in the central repository). For example, you can simply check out a work flow. In that case, you can change that work flow, such as adding a new script to the work flow; however, you cannot change dependent objects in the work flow, such as data flows, and retain the changes in the central repository. Changes to dependent objects will only be retained in the local repository. Alternatively, you can check out the work flow with all of its dependents. In that case, you can make changes to the work flow or any of its dependents and retain the changes in both central and local repositories.

Generally, it is safest to check out an object with all dependents. When you do this, you prevent others from accidentally changing dependent objects.

2.21.6.3.1.1 To check out a single object

1. Open the central object library.
2. Right-click the object you want to check out.
3. Choose *Check Out* > *Object*.

   Alternatively, you can select the object in the central object library, and click the *Check Out Object* button on the top of the central object library.

   The software copies the most recent version of the selected object from the central repository to your local repository, then marks the object as checked out.

2.21.6.3.1.2 To check out an object and its dependent objects

1. Open the central object library.
2. Right-click the object you want to check out.
3. Choose *Check Out* > *Object and dependents*.

   Alternatively, you can select the object in the central object library, and click the *Check Out Object and dependents* button on the top of the central object library.

SAP Data Services copies the most recent version of the selected object and all of its dependent objects from the central repository and marks these objects as checked out.

If a dependent object is checked out by you or another user, then the software alerts you with a Check Out Alert window, asking to get the latest version of the checked out object.

Related Information

Getting objects [page 820]

2.21.6.3.2 Check out single objects or objects with dependents without replacement

When you check out an object, you can replace the object in your local repository with the latest version from the central repository, or you can leave the current version in your local repository intact.

When you check out an object, SAP Data Services copies the object definition from the central repository and replaces any existing definitions for that object in your local repository.

You can check out objects without replacing the objects in your local repository. For example, suppose you are working in your local repository and you make a change to an object that is not checked out. If you determine that the change improves the design or performance of your application, you will want to include that change in the central repository.

To do this, check out the object without replacing the object in your local repository—the object that you have already improved with a change. Then, check the changed object back into the central repository.

> **Note**
>
> Use caution when checking out objects without replacing the version in your local repository. When you do not replace the version in your local repository, you can lose changes that others have incorporated into those objects.

2.21.6.3.2.1 To check out an object or an object and its dependent objects without replacement

1. Open the central object library.
2. Right-click the object you want to check out and choose *Check Out* > *Object* > *without replacement* to check out the single object or choose *Check Out* > *Object and dependents without replacement* to check out the object and all of its dependent objects.

   SAP Data Services marks all appropriate objects as checked out—in both the object library and in the workspace—but does not copy any objects from the central repository to the local repository.

2.21.6.3.3 Check out objects with filtering

When you check out an object with filtering, the object and all its dependents are checked out.

> **Note**
>
> When you check out objects with filtering, you always replace local versions with the filtered objects from the central repository.

2.21.6.3.3.1 To check out and object and its dependent objects with filtering

1. Open the central object library.
2. Right-click the object you want to check out and choose ▶ *Check Out* ▶ *With filtering* ◀.
3. Complete the filtering windows.
4. Click *Finish* to check out the selected objects.

Related Information

Filtering [page 811]

2.21.6.4 Undoing check out

Occasionally, you may decide that you did not need to check out an object because you made no changes. Or, you may decide that the changes you made to a checked-out object are not useful and you prefer to leave the master copy of the object as is. In these cases, you can undo the check out.

When you undo a check out:

- the object in the central repository remains as it was before the checkout; no changes are made and no additional version is saved in the central repository. Only the object status changes from checked out to available.
- the local version of the object maintains the changes you made. If you want the local object to be an exact copy of the central object, perform a *Get latest* operation on that object.

After you undo a check out, other users can check out and make changes to the object.

2.21.6.4.1 To undo single object check out

1. Open the central object library.
2. Select a checked-out object.
3. Click the *Undo object check out* button. Alternatively, right-click the object and select ▶ *Undo Check Out* ▶ *Object* ◀.

SAP Data Services removes the check-out symbol and makes no changes to the object in the central repository. Any checked-out dependent objects remain checked out.

2.21.6.4.2 To undo check out of an object and its dependents

1. Open the central object library.
2. Select the checked-out object that is the highest level for which you want to undo the check out.
3. Click the *Undo object and dependents check out* button. Alternatively, you can right-click the object in the central object library and select ▶ *Undo Check Out* ▶ *Object and dependents* ◀.

SAP Data Services removes the check-out symbols for the object and any dependent objects that are also checked out. No changes are made to these objects in the central repository.

2.21.6.5 Checking in objects

After you finish making changes to checked out objects, you must check them back into the central repository. Checking in objects creates a new version in the central repository, and allows others to get the changes that you have made. Checking in objects also preserves a copy of the changes for revision control purposes. Later, you can get a particular version of a checked in object and compare it to subsequent changes or even revert to the previous version.

Check in an object when you are done making changes, when others need the object that contains your changes, or when you want to preserve a copy of the object in its present state.

Choose a check-in command based on what you will do to an object.

Related Information

Checking in single objects, objects with dependents [page 817]
Checking in an object with filtering [page 817]

2.21.6.5.1 Checking in single objects, objects with dependents

Just as you can check out a single object or an object with all dependent objects, you can check in a single object or an object with all checked-out dependent objects (as calculated in the local repository).

2.21.6.5.1.1 To check in a single object

1. Open the central object library.
2. Select the object you want to check in.
3. Click *Check in object* button at the top of the central object library.

   Alternatively, you can right-click the object in the central object library and select *Check In* > *Object*.
4. A Check In window opens with a *Comment* box, in which you can enter comments. After entering any comments, click *OK*.

   SAP Data Services copies the object from your local repository to the central repository, and removes the check-out mark.

2.21.6.5.1.2 To check in an object and its dependent objects

1. Open the central object library.
2. Select the highest level object you want to check in.
3. Click *Check in object and dependents* button at the top of the central object library.

   Alternatively, you can right-click the object in the central object library and select *Check In* > *Object* and dependents.
4. A Check In window opens with a *Comment* box, in which you can enter comments. After entering any comments, click *OK*.

   SAP Data Services copies the selected object and all of its dependent objects from your repository to the central repository and removes the check-out mark.

2.21.6.5.2 Checking in an object with filtering

Just as you could check out objects with filtering, you can check in objects with filtering. When you check in an object with filtering, the object and all its dependent objects are checked in.

2.21.6.5.2.1 To check in an object with filtering

1. Open the central object library.
2. Right-click the object you want to check out and choose *Check In* › *With filtering*.
3. A Check In window opens with a *Comment* box, in which you can enter comments. After entering any comments, click *OK*.

   SAP Data Services warns you that you are about to create a new version of the object in the central repository.
4. Click *Yes* to continue with the check in.
5. Complete the filtering windows.
6. Click *Finish* to check in the selected objects.

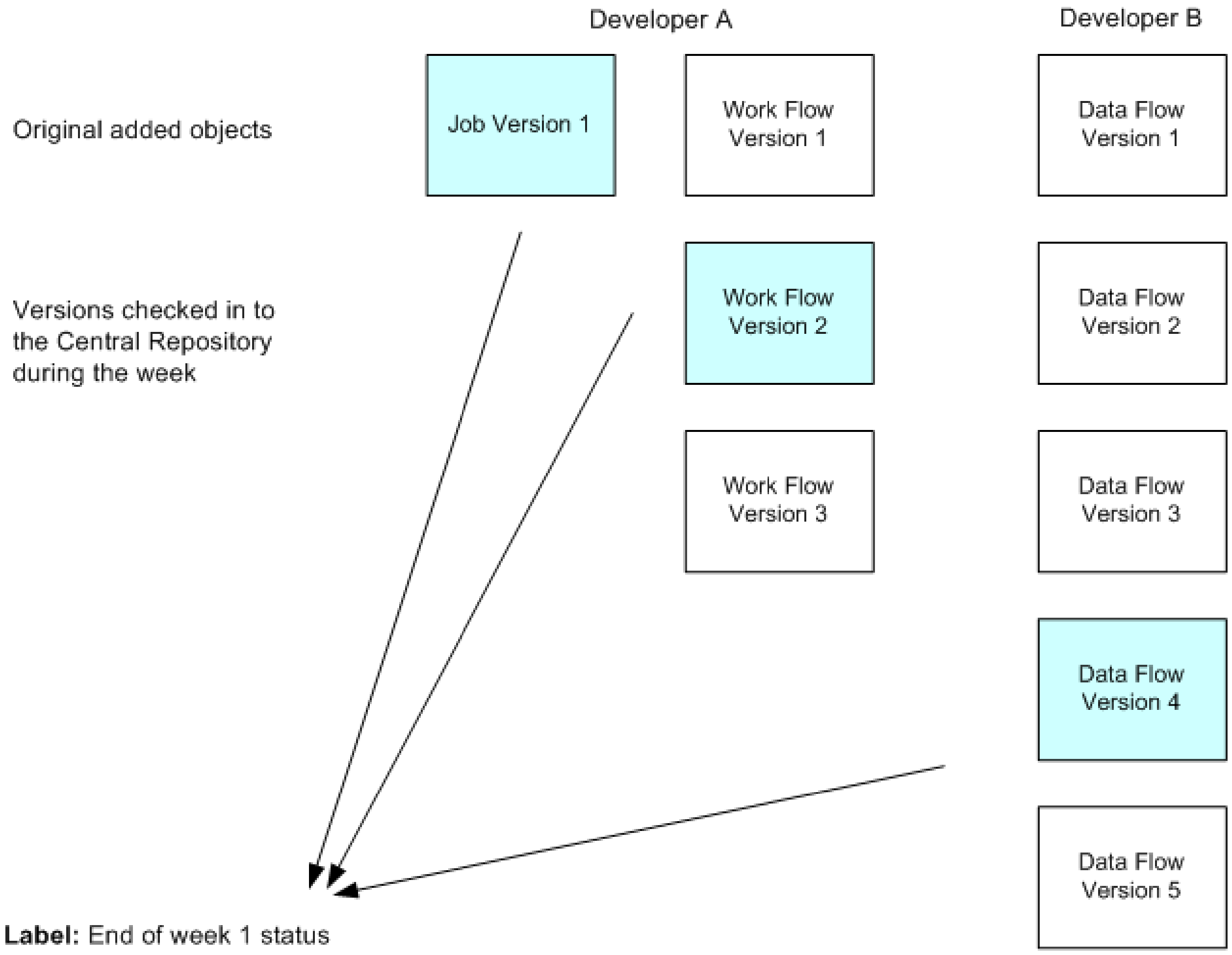
Related Information

Filtering [page 811]

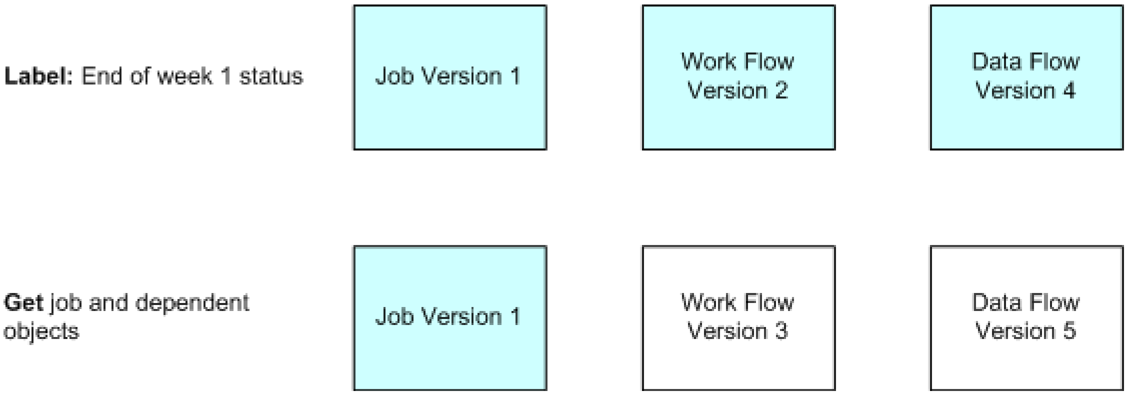
2.21.6.6 Labeling objects

To help organize and track the status of objects in your application, you can label objects. You can choose to either label an object, or label an object and all of its dependent objects. A label not only describes an object, but also allows you to maintain relationships between various versions of objects.

For example, suppose developer A adds a job to the central repository and works on a work flow in that job while developer B works on a data flow in the same job. At the end of the week, after developer A checks in two versions of the work flow and developer B checks in four versions of the data flow to the central repository, the job is labeled "End of week 1 status." This label contains version 1 of the job, version 2 of the work flow, and version 4 of the data flow. Both developers can continue to change their respective work flow and data flow.



At some later point, if you want to get the job with the version of the data flow with this label, getting the job by its label accomplishes this, whereas checking out the job and its dependents does not.



The label "End of week 1 status" serves the purpose of collecting the versions of the work flow and data flow that were checked in at the end of the week. Without this label, you would have to get a particular version of each object in order to reassemble the collection of objects labeled "End of week 1 status."

Related Information

Getting objects [page 820]

2.21.6.6.1 To label an object and its dependents

1. Open the central object library.
2. Right-click the object you want to label and choose *Label Latest Version* > *Object* to label only the highlighted object, or choose *Object and dependents* to label the highlighted object and all its related objects.

   The *Label Latest Version* window opens.
3. In the *Label* box, enter text that describes the current status of the object, then click *OK*.

   The label is inserted in the history of the object and its dependents.

Related Information

Viewing object history [page 822]

2.21.6.6.2 To get a labeled object with filtering

The filtering option for the Get by label operation allows you to filter (selectively change) environment-specific information in object definitions when working in a multi-user environment.

1. Open the central object library.
2. Select the highest level object you want to get.
3. Right-click the object in the central object library and select *Get By Label* > *With filtering* .
4. Complete the filtering window.
5. Click *Finish* to get the selected objects.

Related Information

Filtering [page 811]

2.21.6.7 Getting objects

To make sure that your repository is up-to-date, you "get" objects. When you get an object, you copy the latest version of that object in the central object library and copy it into your local repository, replacing the version in your local repository. When you get an object, you do not check out the object. The object remains free for others to check out and change.

You can get an object with or without dependent objects and filtering.

Related Information

Viewing object history [page 822]

2.21.6.7.1 To get a single object

1. Open the central object library.
2. Select the object you want to get.
3. Click *Get latest version of object* at the top of the central object library.

   Alternatively, right-click the object in the central object library and select ▶ *Get Latest Version* ▶ *Object* ◢.

   The most recent version of the object in the central repository is copied to your local repository.

2.21.6.7.2 To get an object and its dependent objects

1. Open the central object library.
2. Select the highest level object you want to get.
3. Click *Get latest version of objects and dependents* at the top of the central object library.

   Alternatively, right-click the object in the central object library and select ▶ *Get Latest Version* ▶ *Object and dependents* ◢.

   The most recent version of the selected object and all dependent objects from the central repository is copied to your local repository.

2.21.6.7.3 To get an object and its dependent objects with filtering

1. Open the central object library.
2. Select the highest level object you want to get.
3. Right-click the object in the central object library and select ▶ *Get Latest Version* ▶ *With filtering* ◢.
4. Complete the filtering windows.
5. Click *Finish* to get the selected objects.

Related Information

Filtering [page 811]

2.21.6.8 Comparing objects

SAP Data Services allows you to compare two objects from local and central repositories to determine the differences between those objects.

Related Information

Designer Guide: Design and Debug, Comparing Objects [page 714]

2.21.6.9 Viewing object history

The central repository retains a history of all changes made to objects in the central repository. Use this history to help manage and control development of your application.

2.21.6.9.1 To examine the history of an object

1. Open the central object library.
2. Select an object.
3. Click the *Show History* button at the top of the central object library.

   Alternatively, you can right-click the object in the central object library, and choose *Show History*.

   The History window shows several pieces of information about each revision of the object.

Column	Description
Version	The object revision number. Each time a user saves the object, the software creates a new version.
Label	Text that a user enters to describe the status of the object at a given point.
Repository	Information about the local repository from which the software saved this version of the object and the username.
Date	The date and time the software saved this version of the object.

Column	Description
Action	The type of change a user made to the object. This table records actions such as: Checked in — User checked in object
Comment	Comments a user enters when adding an object or checking it into a central repository.

Related Information

Labeling objects [page 818]

2.21.6.9.2 To get a previous version of an object

1. Select an object.
2. Click the *Show History* button at the top of the central object library.
3. Click the version of the object you want.
4. Click the *Get Obj By Version* button.

> i Note
>
> When you get a previous version of an object, you only get the object but not its dependent objects.

2.21.6.9.3 To get an object with a particular label

1. Select an object.
2. Click the *Show History* button at the top of the central object library.
3. Click the version of the object with the particular label you want.
4. Click the *Get By Label* button.

2.21.6.10 Deleting objects

You can delete objects from either the central repository or a local repository. To delete an object from the central repository, right-click the object in the central object library and select *Delete*. To delete an object from the local repository, right-click on the object in the object library and select *Delete*.

When you delete an object from a local repository, you do not automatically delete that object from the active central repository. In fact, you can get the object from the central repository to re-insert it.

Similarly, when you delete an object from a central repository, you do not automatically delete the object from connected local repositories. Until you delete the object from the local repository, you can add the object back to the central repository.

When you delete objects from the central repository, you only delete the selected object and all versions of the selected object; you do not delete any dependent objects.

2.21.7 Migrating Multi-user Jobs

Overview of multi-user job migration

Job migration applies to SAP Data Services on multiple levels: application level, repository management level, and product upgrade level. Application migration is much more flexible in a multi-user environment, allowing you to maintain not only multiple versions of your objects during development, but also during test and production phases if you choose.

Related Information

Application phase management [page 824]
Copying contents between central repositories [page 825]
Central repository migration [page 826]

2.21.7.1 Application phase management

Typically, applications pass through different phases on the way from development to production. For example, an application might pass through three phases:

- Developers creating an application
- Testers validating the application
- Administrators running the application

A single central repository can support your application through all phases. Use job labeling and projects to maintain application components independently for each phase. For example, if development wants to make a certain version of an application ready for testing, they may label it “APPL_V1”. Testers can then get that particular application version using the label and proceed with testing. If testing is successful, an administrator can get the application to run in the production environment. In addition, datastore configurations and file locations allows you to configure the application to run in each local environment.

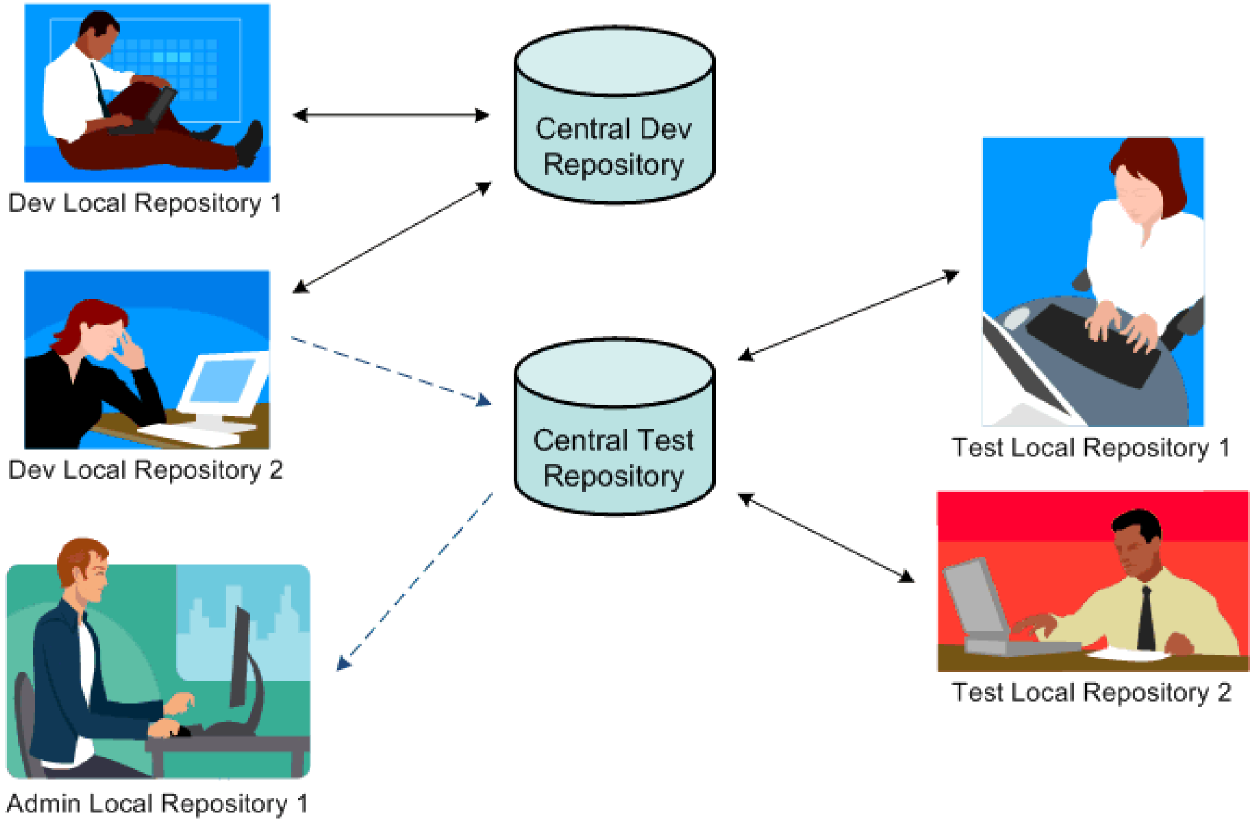
In some situations, you may require more than one central repository for application phase management. If you choose to support multiple central repositories, use a single local repository as a staging location for the transition.

In some situations, you may require more than one central repository for application phase management. Following the example above, once developers create an application version ready for testing by labeling it, a tester would get that version from the development central repository, test it and then check it into a test central repository.

That test central repository will contain all versions tested over time, allowing flexibility for testers to go back to any previous version without relying on the development environment. When an application version passes testing, an administrator can get it from the test repository and make it available in production. Again, if you need to maintain previous versions of an application already in production, you can create another central repository.

With this scheme, a developer will never interfere with the test environment, and a tester will never interfere with a production environment, creating an extremely safe process of migration.

Note that if you choose to support multiple central repositories, use a single local repository as a staging location for file transition.



2.21.7.2 Copying contents between central repositories

You cannot directly copy the contents of one central repository to another central repository. Rather, you must use your local repository as an intermediate repository.

2.21.7.2.1 To copy the contents of one central repository to another central repository

1. Activate the central repository whose contents you will copy.
2. Get the latest version of all objects in this active central repository so they exist in your local repository.
3. Activate the central repository into which you want to copy the contents.
4. The first time you copy the contents, add the objects from your local repository into this central repository.

   However, if you must re-copy the contents of one central repository into another (for example, during your testing phase some part of a job was reassigned to the development phase for redesign), the process is slightly more complex:

   a) First check out specific objects without replacement from the second central repository.
   b) From your local repository, get the latest version of the objects from the first (for example, development) central repository.
   c) Then, instead of adding, check in the updated objects from your local repository to the second (for example, test) central repository.

Related Information

Activating a central repository [page 803]
Getting objects [page 820]
Adding objects to the central repository [page 811]
Checking out objects [page 812]
Checking in objects [page 816]

2.21.7.3 Central repository migration

When you upgrade your version of SAP Data Services, you should migrate your central repository to the new version. It is recommended that you consider the following guidelines when migrating a central repository to a new release of the software.

1. Back up all central repository (as well as local repository) database tables and associated data before upgrading.
2. Maintain a separate central repository for each version of SAP Data Services to preserve object history.
   To preserve the current version and history of objects in your central repository, create a new central repository of your current version of the software and copy the contents of the original central repository to the newly-created one. This way, the second central repository acts as a backup for your objects and associated history information from the older version of the software.
   When you install the new version of the software, upgrade the newly-created central repository to the latest version of the software.
3. Coordinate efforts to upgrade your central repositories and local repositories at the same time.
   Different versions of your central and local repository may not work together. You cannot perform a multi-user operation between a local and central repository of a different software version.

4. Check in all objects (or undo check-outs if objects were not modified after they were checked out) before migrating the central repositories.
   If you cannot upgrade your central and local repositories at the same time, you should check in all objects (or undo check-outs if objects were not modified during check-out), especially those objects checked out to a local repository you will not be immediately upgrading. After you upgrade your central repository to the new version, you will not be able to check in objects from the local repository of the older version of the software.

Related Information

Copying contents between central repositories [page 825]

3 Reference Guide

3.1 Introduction

3.1.1 Welcome to SAP Data Services

3.1.1.1 Welcome

SAP Data Services delivers a single enterprise-class solution for data integration, data quality, data profiling, and text data processing that allows you to integrate, transform, improve, and deliver trusted data to critical business processes. It provides one development UI, metadata repository, data connectivity layer, run-time environment, and management console—enabling IT organizations to lower total cost of ownership and accelerate time to value. With SAP Data Services, IT organizations can maximize operational efficiency with a single solution to improve data quality and gain access to heterogeneous sources and applications.

3.1.1.2 Documentation set for SAP Data Services

You should become familiar with all the pieces of documentation that relate to your SAP Data Services product. The latest Data Services documentation can be found on the *SAP Help Portal*.

Document	What this document provides
Adapter SDK Guide	Information about installing, configuring, and running the Data Services Adapter SDK
Administrator Guide	Information about administrative tasks such as monitoring, lifecycle management, security, and so on.
Customer Issues Fixed	Information about customer issues fixed in this release. ℹ Note In some releases, this information is displayed the Release Notes.
Designer Guide	Information about how to use Data Services Designer.
Documentation Map	Information about available Data Services books, languages, and locations.
Installation Guide for Windows	Information about and procedures for installing Data Services in a Windows environment.
Installation Guide for UNIX	Information about and procedures for installing Data Services in a UNIX environment.
Integrator Guide	Information for third-party developers to access Data Services functionality using web services and APIs.

Document	What this document provides
Master Guide	Information about the application, its components and scenarios for planning and designing your system landscape. Information about SAP Information Steward is also provided in this guide.
Management Console Guide	Information about how to use Data Services Administrator and Data Services Metadata Reports.
Performance Optimization Guide	Information about how to improve the performance of Data Services.
Reference Guide	Detailed reference material for Data Services Designer.
Release Notes	Important information you need before installing and deploying this version of Data Services.
Technical Manuals	A compiled, searchable, “master” PDF of core Data Services books: ● *Administrator Guide* ● *Designer Guide* ● *Reference Guide* ● *Management Console Guide* ● *Performance Optimization Guide* ● *Integrator Guide* ● *Supplement for J.D. Edwards* ● *Supplement for Oracle Applications* ● *Supplement for PeopleSoft* ● *Supplement for Salesforce.com* ● *Supplement for Siebel* ● *Supplement for SAP* ● *Workbench Guide*
Text Data Processing Extraction Customization Guide	Information about building dictionaries and extraction rules to create your own extraction patterns to use with Text Data Processing transforms.
Text Data Processing Language Reference Guide	Information about the linguistic analysis and extraction processing features that the Text Data Processing component provides, as well as a reference section for each language supported.
Tutorial	A step-by-step introduction to using Data Services.
Upgrade Guide	Information to help you upgrade from previous releases of Data Services and release-specific product behavior changes from earlier versions of Data Services to the latest release.
What's New	Highlights of new key features in this SAP Data Services release. This document is not updated for support package or patch releases.
Workbench Guide	Provides users with information about how to use the Workbench to migrate data and database schema information between different database systems.

In addition, you may need to refer to several Supplemental Guides.

Document	What this document provides
Supplement for SAP	Information about interfaces between Data Services, SAP Applications, SAP Master Data Services, SAP NetWeaver BW, and SAP Master Data Services.
Supplement for SuccessFactors	Information about interfaces between Data Services and SuccessFactors.
Supplement for Salesforce.com	Information about how to install, configure, and use the SAP Data Services Salesforce.com Adapter Interface.
Supplement for J.D. Edwards	Information about interfaces between Data Services and J.D. Edwards World and J.D. Edwards OneWorld.
Supplement for Oracle Applications	Information about the interface between Data Services and Oracle Applications.
Supplement for PeopleSoft	Information about interfaces between Data Servicesand PeopleSoft.
Supplement for Siebel	Information about the interface between Data Services and Siebel.

We also include these manuals for information about SAP BusinessObjects Information platform services.

Document	What this document provides
Information platform services Administrator Guide	Information for administrators who are responsible for configuring, managing, and maintaining an Information platform services installation.
Information platform services Installation Guide for UNIX	Installation procedures for SAP BusinessObjects Information platform services on a UNIX environment.
Information platform services Installation Guide for Windows	Installation procedures for SAP BusinessObjects Information platform services on a Windows environment.

3.1.1.3 Accessing documentation

You can access the complete documentation set for SAP Data Services in several places.

3.1.1.3.1 Accessing documentation on Windows

After you install SAP Data Services, you can access the documentation from the Start menu.

1. Choose *Start* > *Programs* > *SAP Data Services 4.2* > *Data Services Documentation* > *All Guides* .
2. Click the appropriate shortcut for the document that you want to view.

3.1.1.3.2 Accessing documentation on UNIX

After you install SAP Data Services, you can access the documentation by going to the directory where the printable PDF files were installed.

1. Go to **<LINK_DIR>**`/doc/book/en/`.
2. Using Adobe Reader, open the PDF file of the document that you want to view.

3.1.1.3.3 Accessing documentation from the Web

You can access the complete documentation set for SAP Data Services from the SAP Business Users Support site.

To do this, go to *http://help.sap.com/bods*.

You can view the PDFs online or save them to your computer.

3.1.1.4 SAP information resources

A global network of SAP technology experts provides customer support, education, and consulting to ensure maximum information management benefit to your business.

Useful addresses at a glance:

Address	Content
Customer Support, Consulting, and Education services *http://service.sap.com/*	Information about SAP support programs, as well as links to technical articles, downloads, and online forums. Consulting services can provide you with information about how SAP can help maximize your information management investment. Education services can provide information about training options and modules. From traditional classroom learning to targeted e-learning seminars, SAP can offer a training package to suit your learning needs and preferred learning style.
Product documentation *http://help.sap.com/bods/*	SAP product documentation.
Supported Platforms (Product Availability Matrix) *https://service.sap.com/PAM*	Get information about supported platforms for SAP Data Services. Use the search function to search for Data Services. Click the link for the version of Data Services you are searching for.
SAP Data Services Community Network *http://scn.sap.com/community/data-services*	Get online and timely information about SAP Data Services, including forums, tips and tricks, additional downloads, samples, and much more. All content is to and from the community, so feel free to join in and contact us if you have a submission.
Blueprints *http://scn.sap.com/docs/DOC-8820*	Blueprints for you to download and modify to fit your needs. Each blueprint contains the necessary SAP Data Services project, jobs, data flows, file formats, sample data, template

Address	Content
	tables, and custom functions to run the data flows in your environment with only a few modifications.
SAPTerm *https://portal.wdf.sap.corp/go/sapterm*	SAP's terminology database, the central repository for defining and standardizing the use of specialist terms.

3.1.2 Overview of this guide

3.1.2.1 About this guide

The *Data Services Reference Guide* provides detailed information about the objects, data types, transforms, and functions in the Designer.

For source-specific information, such as information pertaining to a particular back-office application, refer to the supplement for that application.

3.1.2.2 Who should read this guide

This and other SAP Data Services software documentation assume the following:

- You are a software developer, consultant, or database administrator working on data extraction, data warehousing, data integration, or data quality.
- You understand your source and target data systems, DBMS, legacy systems, business intelligence, and messaging concepts.
- You understand your organization's data needs.
- You are familiar with SQL (Structured Query Language).
- If you are interested in using this software to design real-time processing, you are familiar with:
  - DTD and XML Schema formats for XML files
  - Publishing Web Services (WSDL, HTTP/S and SOAP protocols, and so on.)
- You are familiar with the installation environments: Microsoft Windows or UNIX.

3.2 Objects

This section provides a reference of detailed information about the objects, data types, transforms, and functions in the Designer.

> Note
>
> For information about source-specific objects, consult the reference section in the supplement document for that source.

Related Information

Characteristics of objects [page 833]
Descriptions of objects [page 835]

3.2.1 Characteristics of objects

This section discusses common characteristics of all the objects.

Related Information

Object classes [page 833]
Object options, properties, and attributes [page 834]

3.2.1.1 Object classes

An object's class determines how you create and retrieve the object. There are two classes of objects:

- Reusable objects
- Single-use objects

3.2.1.1.1 Reusable objects

After you define and save a reusable object, SAP Data Services stores the definition in the repository. You can then reuse the definition as often as necessary by creating calls to the definition.

Most objects created in the software are available for reuse. You access reusable objects through the object library.

A reusable object has a single definition; all calls to the object refer to that definition. If you change the definition of the object in one place, and then save the object, the change is reflected to all other calls to the object.

A data flow, for example, is a reusable object. Multiple jobs, such as a weekly load job and a daily load job, can call the same data flow. If the data flow is changed, both jobs call the new version of the data flow.

When you drag and drop an object from the object library, you are creating a new reference (or **`<call>`**) to the existing object definition.

You can edit reusable objects at any time independent of the current open project. For example, if you open a new project, you can go to the object library, open a data flow, and edit it. The object will remain "dirty" (that is, your edited changes will not be saved) until you explicitly save it.

Functions are reusable objects that are not available in the object library. The software provides access to these objects through the function wizard wherever they can be used.

Some objects in the object library are not reusable in all instances:

- Datastores are in the object library because they are a method for categorizing and accessing external metadata.
- Built-in transforms are "reusable" in that every time you drop a transform, a new instance of the transform is created.

"Saving" a reusable object means storing the language that describes the object to the repository. The description of a reusable object includes these components:

- Properties of the object
- Options for the object
- Calls this object makes to other objects
- Definition of single-use objects called by this object

If an object contains a call to another reusable object, only the call to the second object is saved, not changes to that object's definition.

The description is stored even if the object is not successfully validated.

Objects are saved without prompting you:

- When you import an object into the repository.
- When you finish editing:
  - Datastores
  - Flat file formats
  - XML Schema or DTD formats

You can explicitly save the reusable object currently open in the workspace by choosing *Save* from the *Project* menu. If a single-use object is open in the workspace, the *Save* command is not available.

To save all objects in the repository that have changes, choose *Save All* from the *Project* menu.

You are prompted to save all objects that have changes when you execute a job and when you exit the Designer.

3.2.1.1.2 Single-use objects

Single-use objects appear only as components of other objects. They operate only in the context in which they were created.

"Saving" a single-use object means storing the language that describes the object to the repository. The description of a single-use object can only be saved as part of the reusable object that calls the single-use object.

The description is stored even if the object does not validate.

3.2.1.2 Object options, properties, and attributes

Each object is associated with a set of options, properties, and attributes:

- **Options**: Controls the operation of an object. For example, in a datastore, an option is the name of the database to which the datastore connects.
- **Properties**: Documents an object. For example, properties include the name, description of an object, and the date on which it was created. Properties merely describe an object; they do not affect an object's operation.
  To view properties, right-click an object and select Properties.
- **Attributes**: Provides additional information about an object. Attribute values may also affect an object's behavior.
  To view attributes, double-click an object from an editor and click the Attributes tab.

3.2.2 Descriptions of objects

This section describes each object and tells you how to access that object.

The following table lists the names and descriptions of the objects.

Object	Class	Description
Annotation	Single-use	Describes a flow, part of a flow, or a diagram in the workspace.
Catch	Single-use	Specifies the steps to execute if anerror occurs in a given exception group while a job is running.
COBOL copybook file format	Reusable	Defines the format for a COBOL copybook file source.
Conditional	Single-use	Specifies the steps to execute based on the result of a condition.
Batch Job	Reusable	Defines activities that the software executes at a given time including error, monitor and trace messages. Jobs can be dropped only in the project tree. The object created is a direct reference to the object in the object library. Only one reference to a job can exist in the project tree at one time.
Data flow	Reusable	Specifies the requirements for extracting, transforming, and loading data from sources to targets.
Datastore	Single-use	Specifies the connection information needed to access a database or other data source. Cannot be dropped.
Document	Reusable	Available in certain adapter datastores, documents are data structures that can support complicated nested schemas.
DTD	Reusable	A description of an XML file or message. Indicates the format an XML document reads or writes.
Excel workbook format	Reusable	Defines the format for an Excel workbook source.

Object	Class	Description
File format	Reusable	Indicates how flat file data is arranged in a source or target file.
Function	Reusable	Returns a value.
Log	Single-use	Records information about a particular execution of a single job.
Message function	Reusable	Available in certain adapter datastores, message functions can accommodate XML messages when properly configured.
Outbound message	Reusable	Available in certain adapter datastores, outbound messages are XML-based, hierarchical communications that real-time jobs can publish to adapters.
Project	Single-use	Groups jobs for convenient access.
Query transform	Single-use	Retrieves a data set that satisfies conditions that you specify.
Real-time job	Reusable	Defines activities that the software executes on-demand. Real-time jobs are created in the Designer, then configured and run as services associated with an Access Server in the Administrator. Real-time jobs are designed according to data flow model rules and run as a request-response system.
Script	Single-use	Evaluates expressions, calls functions, and assigns values to variables.
Source	Single-use	An object from which the software reads data in a data flow.
Table	Reusable	Indicates an external DBMS table for which metadata has been imported, or the target table into which data is or has been placed. A table is associated with its datastore; it does not exist independently of a datastore connection. A table retrieves or stores data based on the schema of the table definition from which it was created.
Target	Single-use	An object in which the software loads extracted and transformed data in a data flow.
Template table	Reusable	A new table you want added to a database. All datastores except SAP datastores have a default template that you can use to create any number of tables in the datastore. The software creates the schema for each instance of a template table at runtime. The created schema is based on the data loaded into the template table.
Transform	Reusable	Performs operations on data sets.

Object	Class	Description
		Requires zero or more data sets; produces zero or one data set (which may be split).
Try	Single-use	Introduces a try/catch block.
While loop	Single-use	Repeats a sequence of steps as long as a condition is true.
Work flow	Reusable	Orders data flows and operations supporting data flows.
XML file	Single-use	A batch or real-time source or target. As a source, an XML file translates incoming XML-formatted data into data that the software can process. As a target, an XML file translates the data produced by a data flow, including nested data, into an XML-formatted file.
XML message	Single-use	A real-time source or target. As sources, XML messages translate incoming XML-formatted requests into data that a real-time job can process. As targets, XML messages translate the result of the real-time job, including hierarchical data, into an XML-formatted response and sends the messages to the Access Server.
XML Schema	Reusable	A description of an XML file or message. Indicates the format an XML document reads or writes.
XML template	Single-use	A target that creates an XML file that matches a particular input schema. No DTD or XML Schema is required.

3.2.2.1 Annotation

Class

Single-use

Access

Click the annotation icon in the tool palette, then click in the workspace.

Description

Annotations describe a flow, part of a flow, or a diagram in a workspace. An annotation is associated with the job., work flow, or data flow where it appears. When you import or export that job, work flow, or data flow, you import or export associated annotations.

> **Note**
>
> An annotation has no options or properties.

Related Information

Designer Guide: Creating annotations [page 194]

3.2.2.2 Batch Job

Class

Reusable

Access

- In the object library, click the *Jobs* tab.
- In the project area, select a project and right-click *Batch Job*.

Description

A batch job is a set of objects that you can schedule and execute together. To execute the steps of any object, the object must be part of a job.

A batch job can contain the following objects:

- Data flows

    - Sources
    - Transforms
    - Targets
- Work flows
- Scripts
- Conditionals
- Try/catch blocks
- While Loops

You can run batch jobs such that you can automatically recover from jobs that do not execute successfully. During automatic recovery, SAP Data Services retrieves the results from steps that were successfully completed in the previous run and executes all other steps. Specifically, the software retrieves results from the following types of steps:

- Work flows
- Data flows
- Script statements
- Custom functions (stateless type only)
- SQL function
- EXEC function
- get_env function
- rand function
- sysdate function
- systime function

Batch jobs have the following built-in attributes:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the job.
Date created	The date when the object was created.

Batch and real-time jobs have properties that determine the information collected and logged when running the job. You can set the default properties that apply each time you run the job or you can set execution (run-time) properties that apply for a particular run. Execution properties override default properties.

To set default properties, select the job in the project area or the object library, right-click, and choose *Properties* to open the Properties window.

Execution properties are set as you run a job. To set execution properties, right-click the job in the project area and choose *Execute*. The Designer validates the job and opens the Execution Properties window.

You can set several types of execution properties:

- Execution options
- Trace properties
- Global variables
- Substitution parameters

Related Information

Designer Guide: Setting global variable values [page 408]

3.2.2.2.1 Execution Options

Use execution options to help capture and diagnose errors using log, auditing, statistics collection, or recovery options.

Log information goes to one of three files (in the **`<DS_COMMON_DIR>`**`\log\`**`<Job_Server_Name>`**`\`**`<repository name>`** directory):

- Monitor log file
- Trace log file
- Error log file

When you execute a job, you can also select a system configuration and a Job Server or server group from the *Execution Options* tab of the Execution Properties window.

Select the *Execution Options* tab to set the following options.

Options	Description
Collect statistics for monitoring	Select this check box if you want to display cache statistics in the Performance Monitor in the Administrator. (The default is cleared.) Note Use this option only if you want to look at the cache size. For more information, see the Using Caches section in the *Performance Optimization Guide*.
Collect statistics for optimization	Select this check box if you want to collect statistics that the Data Services optimizer will use to choose an optimal cache type (in-memory or pageable). This option is not selected by default. For more information, see Caching sources in the *Performance Optimization Guide*. For more information, see the Using Caches section in the *Performance Optimization Guide*.
Disable data validation statistics collection	Select this check box if you do not want to collect data validation statistics for any validation transforms in this job. (The default is cleared.) For more information about data validation statistics, see Data Validation dashboards Settings control panel" in the *Management Console Guide*. For more information about data validation statistics, see the Data Validation Dashboard Reports section in the *Management Console Guide*.
Distribution level	Select the level within a job that you want to distribute to multiple job servers for processing:

Options	Description
	- Job - The whole job will execute on an available Job Server. - Data flow - Each data flow within the job can execute on an available Job Server. - Sub data flow - Each sub data flow (can be a separate transform or function) within a data flow can execute on an available Job Server. For more information, see Using grid computing to distribute data flows execution in the *Performance Optimization Guide*. For more information, see the Distributing Data Flow Execution section in the *Performance Optimization Guide*.
Enable auditing	Clear this check box if you do not want to collect audit statistics for this specific job execution. (The default is selected.) For more information about auditing, see Using Auditing in the *Designer Guide*. For more information about auditing, see the Data Assessment section in the *Designer Guide*.
Enable recovery	(Batch jobs only) Select this check box to enable the automatic recovery feature. When enabled, the software saves the results from completed steps and allows you to resume failed jobs. You cannot enable the automatic recovery feature when executing a job in data scan mode. See Automatically recovering jobs in the *Designer Guide* for information about the recovery options. See the Recovery Mechanisms section in the *Designer Guide* for information about the recovery options. See “Loading Big Data file with Recovery option” in the *Designer Guide* for information about big data file recovery options. This property is only available as a run-time property. It is not available as a default property.
Export Data Quality reports	Generates and exports all specified job reports to the location specified in the Report Server Configuration node in the Management Console Administrator. By default, the reports are exported to `<DS_COMMON_DIR>\DataQuality\reports\<repository\job>`.
Job Server or Server group	Select the Job Server or server group to execute this job. A Job Server is defined by a host name and port while a server group is defined by its name. The list contains Job Servers and server groups linked to the job's repository. For an introduction to server groups, see Server group architecture in the *Management Console Guide*. For an introduction to server groups, see the Server Groups section in the *Management Console Guide*. When selecting a Job Server or server group, remember that many objects in the Designer have options set relative to the Job Server's location. For example: - Directory and file names for source and target files - Bulk load directories

Options	Description
Monitor sample rate (# of seconds)	Enter the number of seconds elapsed before the software writes information to the monitor log file and updates job events. The software writes information about the status of each source, target, or transform. For example, if you enter 30, the logs update every 30 seconds. The default is 5 seconds. When setting the value, you must evaluate performance improvements gained by making fewer calls to the operating system against your ability to find errors quickly. With a higher monitor sample rate, more data collects before calling the operating system to open the file: performance improves. However, with a higher monitor rate, more time passes before you are able to see any errors. Setting the rate to 0 disables the monitoring feature. i Note If you use a virus scanner on your files, exclude the log from the virus scan. Otherwise, the virus scan analyzes the log repeated during the job execution, which causes a performance degradation.
Print all trace messages	Select this check box to print all trace messages to the trace log file for the current Job Server. Selecting this option overrides the trace properties set on the *Trace* tab.
Recover from last failed execution	(Batch Job only) Select this check box to resume a failed job. The software retrieves the results from any steps that were previously executed successfully and re-executes any other steps. This option is a run-time property. This option is not available when a job has not yet been executed or when recovery mode was disabled during the previous run.
System configuration	Select the system configuration to use when executing this job. A system configuration defines a set of datastore configurations, which define the datastore connections. For more information, see Creating and managing multiple datastore configurations in the *Designer Guide*. For more information, see the Datastores section in the *Designer Guide*. If a system configuration is not specified, the software uses the default datastore configuration for each datastore. This option is a run-time property. This option is only available if there are system configurations defined in the repository.
Use collected statistics	Select this check box if you want the optimizer to use the cache statistics collected on a previous execution of the job. (The default is selected.) For more information, see Monitoring and tuning caches in the *Performance Optimization Guide*. For more information, see section 6: Using Caches in the *Performance Optimization Guide*.

3.2.2.2.2 Trace properties

Use trace properties to select the information that SAP Data Services monitors and writes to the trace log file during a job. The software writes trace messages to the trace log associated with the current Job Server and writes error messages to the error log associated with the current Job Server.

To set trace properties, click the Trace tab. To turn a trace on, select the trace, click Yes in the Value list, and click OK. To turn a trace off, select the trace, click No in the Value list, and click OK.

You can turn several traces on and off.

Trace	Description
Access Server Communication	Writes messages exchanged between the Access Server and a service provider, including: • The registration message, which tells the Access Server that the service provider is ready • The request the Access Server sends to the service to execute • The response from the service to the Access Server • Any request from the Access Server to shut down
Assemblers	Writes messages for Substitution Parameter and SDK Transforms: • Substitution parameters - Writes trace messages such as substitution configuration used and the actual value substituted for each substitution parameter. • SDK transforms - Writes transform-specific information specified in the form of XML. This information can be hierarchical. At runtime this XML information is extracted or assembled. The trace messages write the assembled XML.
Audit Data	Writes a message when auditing: • Collects a statistic at an audit point • Determines if an audit rule passes or fails
Data Flow	Writes a message when the data flow starts, when the data flow successfully finishes, or when the data flow terminates due to error. This trace also reports when the bulk loader starts, any bulk loader warnings occur, and when the bulk loader successfully completes.
IDoc file reader	Writes a message when reading IDoc files including: • start reading the IDoc file • stop reading the IDoc file • result of the IDoc file validation
Memory Source	Writes a message for every row retrieved from the memory table.
Memory Target	Writes a message for every row inserted into the memory table.
Optimized data flow	For consulting and customer assurance use.

Trace	Description
RFC Function	Writes a message related to RFC calls including: ● start of RFC call ● end of RFC call ● a message for each record received from the software for the RFC call
Row	Writes a message when a transform imports or exports a row.
SAP Table Reader	Writes messages from readers of SAP system tables including: ● start reading from table ● stop reading from table ● start of connection to SAP system where table is present ● end of connection to SAP system
Scripts and Script Functions	Writes a message when the software runs a script or invokes a script function. Specifically, this trace links a message when: ● The script is called. Scripts can be started any level from the job level down to the data flow level. Additional (and separate) notation is made when a script is called from within another script. ● A function is called by the script. ● The script successfully completes.
Session	Writes a message when the job description is read from the repository, when the job is optimized, and when the job runs.
SQL Functions	Writes data retrieved before SQL functions: ● Every row retrieved by the named query before the SQL is submitted in the key_generation function ● Every row retrieved by the named query before the SQL is submitted in the lookup function (but only if PRE_LOAD_CACHE is not specified). ● When mail is sent using the mail_to function.
SQL Loaders	Writes a message when the bulk loader: ● Starts ● Submits a warning message ● Completes successfully ● Completes unsuccessfully, if the *Clean up bulk loader directory after load* option is selected Additionally, for Microsoft SQL Server and SAP SAP Sybase ASE, writes when the SQL Server bulk loader: ● Completes a successful row submission ● Encounters an error This instance reports all SQL that the software submits to the target database, including:

Trace	Description
	- When a `truncate table` command executes if the *Delete data from table before loading* option is selected. - Any parameters included in PRE-LOAD SQL commands - Before a batch of SQL statements is submitted - When a template table is created (and also dropped, if the *Drop/Create* option is turned on) - When a `delete from table` command executes if auto correct is turned on (Informix environment only). This trace also writes all rows that the software loads into the target.
SQL Only	Use in conjunction with Trace the SQL Transforms option, the Trace SQL Readers option, or the Trace SQL Loaders option to stop the writing of trace messages for data sent and received from the database.
SQL Readers	Writes the SQL query block that a script, Query transform, or SQL function submits to the system. Also writes the SQL results.
SQL Transforms	Writes a message (using the Table_Comparison transform) about whether a row exists in the target table that corresponds to an input row from the source table. The trace message occurs before submitting the query against the target and for every row retrieved when the named query is submitted (but only if caching is not turned on).
Stored Procedure	Writes a message when the software invokes a stored procedure. This trace reports: - When the stored procedure starts - The SQL query submitted for the stored procedure call - The value (or values) of the input parameter (or parameters) - The value (or values) of the output parameter (or parameters) - The return value (if the stored procedure is a stored function) - When the stored procedure finishes
Tables	Writes a message when a table is created or dropped. The message indicates the datastore to which the created table belongs and the SQL statement used to create the table.
Trace ABAP	Writes a message when an ABAP data flow starts or stops, and to report the ABAP job status.
Trace Parallel Execution	Writes messages describing how data in a data flow is parallel processed.
Transform	Writes a message when a transform starts, completes, or terminates.
Work Flow	Writes a message when the work flow description is read from the repository, when the work flow is optimized, when the work flow runs, and when the work flow ends.

3.2.2.3 Catch

Class

Single-use

Access

With a work flow in the workspace, click the catch icon in the tool palette.

Description

A catch object is part of a serial sequence called a try/catch block. The try/catch block allows you to specify alternative work flows if one or more errors occur while executing a job. Try/catch blocks "catch" exception groups of errors, apply solutions that you provide, and continue execution.

For each catch object in the try/catch block, specify the following:

- One or more groups of exceptions that the catch object handles.
  If you want to assign different actions to different exception groups, add a catch for each set of actions.
- The actions to execute if an exception in the indicated exception groups occurs.
  It is recommended that you define, test, and save the actions as a separate object rather than constructing them inside the catch editor. The actions can be a single script object, a data flow, a work flow, or a combination of these objects.
- Optional error functions inside the catch block to identify details of the error.

If an exception is thrown during the execution of a try/catch block, and if no catch object is looking for that exception group, then the exception is handled by normal error logic.

For batch jobs only, do not reference output variables from a try/catch block in any subsequent steps if you are using the automatic recovery feature. Referencing such variables could alter the results during automatic recovery.

Also, try/catch blocks can be used within any real-time job component. However, try/catch blocks cannot straddle a real-time processing loop and the initialization or cleanup component of a real-time job.

Catch objects have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

The following table describes exception groups that you can catch in a try/catch block:

Exception group	Group number	Description
Catch All Exceptions	All	All errors
Execution errors	1001	Errors in the job server
Database access errors	1002	Errors from the database server while reading data, writing data, or bulk loading to tables
Database connection errors	1003	Errors connecting to database servers
Flat file processing errors	1004	Errors processing flat files
File access errors	1005	Errors accessing local and FTP files
Repository access errors	1006	Errors accessing the repository
SAP Execution errors	1007	Errors from the SAP system
System resource exception	1008	Errors accessing operating system resources
SAP BW execution errors	1009	Errors from the SAP BW system
XML processing errors	1010	Errors processing XML files and messages
COBOL copybook errors	1011	Errors processing COBOL copybook files
Excel book errors	1012	Errors processing Excel books
Data Quality transform errors	1013	Errors processing Data Quality transforms

3.2.2.3.1 Catch error functions

Syntax

The following table describes error functions that you can use in the script that your catch work flow executes.

i Note

You can only invoke these error functions inside a catch script, a user function, or in an audit script for a data flow. If you call these error functions in any other place, a validation error occurs.

Catch error function	Return data type and size	Description
error_timestamp()	timestamp	Returns the timestamp of the caught exception.

Catch error function	Return data type and size	Description
error_context()	varchar 512	Returns the context of the caught exception. For example, *|Session datapreview_job|Dataflow debug_DataFlow|Transform Debug*
error_message()	varchar 512	Returns the error message of the caught exception.
error_number()	int	Returns the error number of the caught exception.

Related Information

Designer Guide: Work flows, Example: Catching details of an error [page 340]

3.2.2.3.2 Catch scripts

A script is the most common action that a catch executes for a thrown exception. The catch script can contain the following:

- Catch error functions and other function calls
- Nested try/catch blocks
- if statements to perform different actions for different exceptions

Syntax

The syntax for a try/catch block within a script is as follows:

```
try
begin
   <steps>
end
catch(<integer_constants>)
begin
    <steps>
end
```

Where

`<steps>`	Catch error functions, other function calls, if statements, or other statements you want to perform for an error in the specified exception group number.

`<integer_constants >`	One or more exception group numbers that you want to catch.
	Use a comma to separate exception group numbers. For example,
	`catch(1002,1003)`
	Specify `ALL` to catch all exceptions:
	`catch(ALL)`

Related Information

Designer Guide: Work flows, Example: Catching details of an error [page 340]

3.2.2.4 COBOL copybook file format

Class

Reusable

Access

In the object library, click the *Formats* tab.

Description

A COBOL copybook file format describes the structure defined in a COBOL copybook file (usually denoted with a .cpy extension). You store templates for file formats in the object library. You use the templates to define the file format of a particular source in a data flow.

The following tables describe the *Import*, *Edit*, and Source *COBOL copybook* options.

3.2.2.4.1 Import or Edit COBOL copybook format options

The *Import COBOL Copybook* and *Edit COBOL Copybook* format windows include options on the following tabs:

- Format
- Data File
- Data Access

The *Edit COBOL Copybook* format window includes options on the following tabs:

- Field ID
- Record Length Field

Format tab

The Format tab defines the parameters of the COBOL copybook format. Format tab options are not editable in the *Edit COBOL Copybook* window.

Format option	Description
File name	Type or browse to the COBOL copybook file name (usually has a .cpy extension). This file contains the schema definition. For added flexibility, you can enter a variable for this option.
Expand OCCURS	Specifies the way to handle OCCURS groups. These groups can be imported with each field in an OCCURS group in one of the following ways: • Getting a sequential suffix for each repetition: fieldname_1, fieldname_2, etc. (expanded view) • Appearing only once in the copybook's schema (collapsed view). For a collapsed view, the output schema matches the OCCURS group definition, and for each input record there will be several output records. If a copybook contains more than one OCCURS group, you must select this box. The default is selected.
Ignore redefines	Determines whether or not to ignore REDEFINES clauses.
Source format	The format of the copybook source code. Options include: • *Free*—All characters on the line can contain COBOL source code.

Format option	Description
	- *Smart mode*—the softwareattempts to determine whether the source code is in Standard or Free format; if this does not produce the desired result, choose the appropriate source format (Standard or Free) manually for reimporting. - *Standard*—The traditional (IBM mainframe) COBOL source format, where each line of code is divided into the following five areas: sequence number (1-6), indicator area (7), area A (8-11), area B (12-72) and comments (73-80).
Source codes [start]	Defines the start column of the copybook source file to use during the import. Typical value is 7 for IBM mainframe copybooks (standard source format) and 0 for free format.
Source codes [end]	Defines the end column of the copybook source file to use during the import. Typical value is 72 for IBM mainframe copybooks (standard source format) and 9999 for free format.
Generate record number field	If selected, creates a new field at the beginning of the schema that the software populates at run time with the record number.

Data File tab

The Data File tab defines the parameters of the data file.

Data file option	Description
Directory	Specifies the directory that contains the COBOL copybook data file to import. For added flexibility, you can enter a variable for this option. If you include a directory path here, then enter only the file name in the *Name* field. During design, you can specify a file in one of the following ways: - For a file located on the computer where the Designer runs, you can use the Browse button. - For a file located on the computer where the Job Server runs, you must type the path to the file. You can type an absolute path or a relative path, but the Job Server must be able to access it.
File name	Type the name or browse to the COBOL copybook data file. You can use variables or wild cards (* or ?). If you leave *Directory* blank, then type the full path and file name here.
Type	Specifies the record format—fixed or variable: - *Fixed(F)* - *Variable(V)*
Has record length	Specifies whether variable-length records of the data file contain information about the length of each record. The possible values are: - *2-byte* integer - *2-byte followed by 0x0000* (integer followed by two zero bytes)

Data file option	Description
	- *4-byte* integer - *None*—No length information at the beginning of each record
Record size	Defines fixed record length in bytes. All records in the file have this length (padded, if necessary).
Record trailer length	Specifies the length of extra character padding in bytes at the end of each record.
Has record mark	Defines whether there is an extra byte in the beginning of each record's data.
Integer format	Describes how the existing data file stores binary data: - *Big endian*—the most significant byte comes first - *Little endian*—the least significant byte comes first
Code page	Specifies the character encoding of character data in the data file.
Skip first	Defines the number of data records to skip before starting to process the file. For added flexibility, you can enter a variable for this option.
Read total	Defines the number of records to read and process. For added flexibility, you can enter a variable for this option.
Low Value and High Value	Identifies a valid hexadecimal value for a low value, high value, or both from the copybook and applies the associated *Action*. You can also use a variable to define a different value at run time. The Low Value default is 0x40 and the High Value default is 0xFF. For example, if the source field is binary 0x40, enter a Low Value of 0x40 and select the action *Convert to NULL*. The result would be as follows for these data types: - Char—Character represented by 0x40 - Packed decimal—NULL - Binary—0x40
Action	For a Low or High Value, applies one of the following actions: - *No conversion*—Reads the value as an ASCII character (default). - *Convert to NULL*—Converts the given value to NULL. - *Convert to 0*—Converts the given value to 0.

Data Access tab

The Data Access tab specifies how access the data file. If both check boxes are cleared, the software assumes the data file is on the same computer as the Job Server.

Data access option	Description
FTP	Select to use FTP to access the data file.
Host	Type the computer (host) name, fully qualified domain name, or IP address of the computer where the data file resides.

Data access option	Description
User	Type the FTP user name.
Password	Type the FTP user password.
Directory	Type or browse to the directory that contains the COBOL copybook data file to import. If you include a directory path here, then enter only the file name in the *Name* field.
File name	Type or browse to the COBOL copybook data file *Name*. You can use variables or wild cards (* or ?). If you leave *Directory* blank, then type a full path and file name here.
Custom	Select to use a custom executable to access the data file.
Executable	Type the name of the program to read data file.
User	Type the user name.
Password	Type the password.
Arguments	Include any custom program arguments.

Field ID tab

If the imported copybook contains multiple schemas, the Field ID tab is visible on the *Edit COBOL Copybook* window. These options allow you to create rules for identifying which records represent which schemas.

Field ID option	Description
Use field <FIELD NAME> as ID	Select to set a value for the field selected in the top pane. Clear to not set a value for that field.
Edit	Changes the selected value in the Values pane to editable text.
Delete	Deletes the selected value in the Values pane.
Insert above	Inserts a new value in the Values pane above the selected value.
Insert below	Inserts a new value in the Values pane below the selected value.

Record Length Field tab

After you import a copybook, the Record Length Field tab is visible on the *Edit COBOL Copybook* window. It lets you identify the field that contains the length of the schema's record.

Record Length Field column	Description
Schema	The data schemas in the copybook.

Record Length Field column	Description
Record length field	Click to enable a drop-down menu where you select a field (one per schema) that contains the record's length.
Offset	The value that results in the total record length when added to the value in the Record length field. The default value for offset is 4; however, you can change it to any other numeric value.

3.2.2.4.2 COBOL copybook source options

The source editor includes the following COBOL copybook options on the following tabs:

- Source
- Field clauses
- Data File
- Data Access

Source tab

Source option	Description
Make port	Makes the source table an embedded data flow port. For more information, see the Embedded Data Flows section in the *Designer Guide*.

Table 16: Performance

Source option	Description
Join rank	Indicates the rank of the source relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. For new jobs, specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For more information, see the Other Tuning Techniques section in the *Performance Optimization Guide*.
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join.

Source option	Description
	Options are: - *Yes*: The source is always cached unless it is the outer-most source in a join. - *No*: The source is never cached. The default is *Yes*. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. For new jobs, specify the cache only in the Query transform editor.

Table 17: Error handling

Source option	Description
Log data conversion warnings	Determines whether to include data-type conversion warnings in the error log. Defaults to *Yes*.
Maximum warnings to log	If Log data conversion warnings is enabled, you can limit how many warnings are loged. Defaults to *{no limit}*.

Table 18: Include file name column

Source option	Description
Include file name column	Determines whether to add a column that contains the source file name in the source output. Defaults to *No*. Change the value to *Yes* when you want to identify the source file in situations such as the following: - You specified a wildcard character to read multiple source COBOL copybooks at one time - You load from different source copybooks on different runs
Modify	If the file name is included, this button enables you to modify *File name column* and *Column size*.
File name column	If the file name is included, the name of the column that holds the source file name. Defaults to *DI_FILENAME*.
Column size	If the file name is included, the size (in characters) of the column that holds the source file name. Defaults to *100*. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.
Include path	If the file name is included, determines whether to include the full path name of the source file. Defaults to *No*.

Field clauses tab

The Field clauses tab displays the attributes for a selected column.

Field clauses option	Description
Possible values	Enter values here to force the software to only process rows that contain the specified value(s). Separate multiple values with the pipe character (|). You can click the ellipses button to open the smart editor; for details on how to use the smart editor, see the Smart Editor section in the *Reference Guide*.
Level	The level number (01-50) assigned to the field in the source record definition.
Original name	The name of the field in the copybook.
Original picture	The PICTURE clause of the field in the copybook.
Original usage	The USAGE clause of the field in the copybook.
Min occurs	Minimum number of occurrences for this field (if this field is a part of an OCCURS group).
Max occurs	Maximum number of occurrences for this field (if this field is a part of an OCCURS group).
Occurs depending on	Specifies the repetition counter field name for the ODO (OCCURS DEPENDING ON).
Redefines	Specifies the name of another field that this one REDEFINES.
Sign separate	Specifies whether the sign is stored separately from the field's value.
Sign position	Specifies whether the sign is LEADING or TRAILING.
Multiply by	Specifies whether the field needs to be scaled (multiplied or divided by a certain number). For example, if the field's PICTURE clause is 9(5)P(3), the value of the field from the data file will be multiplied by 1000.

Related Information

Import or Edit COBOL copybook format options [page 850]

3.2.2.5 Conditional

Class

Single-use

Access

With a work flow diagram in the workspace, click the conditional icon in the tool palette.

Description

A conditional implements if/then/else logic in a work flow.

For each conditional, specify the following:

- *If*: A Boolean expression defining the condition to evaluate.
  The expression evaluates to TRUE or FALSE. You can use constants, functions, variables, parameters, and standard operators to construct the expression. For information about expressions, see Chapter 3: Smart Editor.

  > Note
  >
  > Do not put a semicolon (;) at the end of your expression in the If box.

- *Then*: A work flow to execute if the condition is TRUE.
- *Else*: A work flow to execute if the condition is FALSE.
  This branch is optional.

The *Then* and *Else* branches of the conditional can be any steps valid in a work flow, including a call to an existing work flow.

Conditionals have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

Related Information

Reference Guide: Smart Editor [page 1021]

3.2.2.6 Data flow

Class

Reusable

Access

- In the object library, click the *Data Flows* tab.
- With a work flow diagram in the workspace, click the data flow icon in the tool palette.

Description

A data flow extracts, transforms, and loads data.

You can define parameters to pass values into the data flow. You can also define variables for use inside the data flow.

When SAP Data Services executes data flows, it optimizes the extract, transform, and load requirements into commands to the DBMS and commands executed internally. Where it can, the software runs these operations in parallel.

By definition, a data flow can contain the following objects:

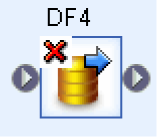
- **Sources**: Files, tables, XMl files, XML messages (real-time jobs only), documents, or pre-defined template tables
- **Targets**: Files, tables, XML files, XML messages (real-time jobs only), outbound messages, documents, XML template, or template tables
- **Transforms**: The Query transform is the most commonly used transform

You can view the SQL code the software generates for table sources in data flows and improve your data flow design accordingly.

Data flows have several built-in properties.

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the data flow.

If you delete a data flow from the object library, calls to the object are replaced with an icon indicating that the calls are no longer valid in the workspace.



Related Information

Performance Optimization Guide: Viewing SQL [page 2121]

3.2.2.6.1 Executing jobs only once

You can ensure that a job executes a data flow only one time by selecting the *Execute only once* check box on the data flow Properties window. When you select this check box, SAP Data Services executes only the first occurrence of the data flow and skips subsequent occurrences of it in the job. You might use this feature when developing complex jobs with multiple paths, such as those containing try/catch blocks or conditionals, and you want to ensure that the software executes a particular data flow only once. Before selecting the *Execute only once*, note that:

- If you design a job to execute the same *Execute only once* data flow in parallel flows, the software only executes the first occurrence of that data flow and you cannot control which one is executed first. Subsequent flows wait until the first one is processed. The engine provides a wait message for each subsequent data flow. Since only one Execute only once data flow can execute in a single job, the engine skips subsequent data flows and generates a second trace message for each, "Data flow n did not run more than one time. It is an execute only once flow."
- The *Execute only once* data flow option does not override the *Recover as a unit* work flow option and the *Enable recovery* job option.

3.2.2.6.2 Parallel processing

You can run certain transforms and functions in parallel by entering a number in the *Degree of parallelism* box on your data flow Properties window. When you drop a transform into the data flow and a function into each transform, the number you enter in the *Degree of parallelism* box is the maximum number of instances that can be generated for each transform or function in the data flow.

Related Information

Performance Optimization Guide: Degree of parallelism [page 2142]

3.2.2.6.3 Caching data

You can cache data to improve performance of operations such as joins, groups, sorts, lookups, and table comparisons. You can select one of the following values for the *Cache type* option on your data flow Properties window:

- In-Memory: Choose this value if your data flow processes a small amount of data that can fit in the available memory.
- Pageable: This value is the default.

  > Note
  >
  > For data flows that you created prior to version 11.7, the default cache type is in-memory. If the data retrieved does not fit into available memory, change the cache type to pageable in the data flow Properties window.

> Note
>
> You cannot use pageable cache with nested data or LONG data types.

Related Information

Performance Optimization Guide: Caching sources [page 2130]

3.2.2.7 Datastore

Class

Reusable

Access

In the object library, click the *Datastores* tab.

Description

A datastore provides a connection to a data source such as a database. Through the datastore connection, SAP Data Services can import descriptions of the data source such as its metadata. When you specify tables as sources or targets in a data flow, the software uses the datastore to determine how to read data from or load data to those tables. In addition, some transforms and functions require a datastore name to qualify the tables they access.

Datastores have the following properties:

Property	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object. You cannot change the name of a datastore after creation.
Description	Text that you enter to describe and document the datastore.
Date_created	The date that you created the datastore. You cannot change this value.

Note

If you delete a datastore from the object library, you must remove references to the datastore from the following locations:

- Source or target tables using this datastore in your diagrams
- The lookup and key_generation functions and Key_Generation, History_Preserving, Table_Comparison, and SQL transform references

3.2.2.7.1 Datastore editor

To open the Datastore Editor, go to the Datastores tab in the object library, right-click the white space, and select *New*. Alternatively, you can right-click the name of an existing datastore and select *Edit*. The Datastore Editor consists of several windows.

- *New* opens the *Create New Datastore* window
- *Edit* opens the Edit Datastore **<DatastoreName>** window

Creating a new Datastore—Basic Configuration Options

Initially only two options appear on this resizable window: *Datastore Name* and *Datastore type*. When you select a datastore type, the window automatically updates to display other options relevant to that type. The combination of *Datastore type* and *Database type* determine the rest of your available options for that datastore.

There are three general categories of datastores:

- Database datastores allow you to connect to supported databases.
- Adapter datastores allow you to connect to adapters.

- Application datastores, such as PeopleSoft and JDE One World allow you to connect to applications that run on databases. You can select these applications by name from the *Datastore type* list.

SAP Data Services supports changed-data capture (CDC) and transfer tables with Oracle databases. So in this case, the Designer displays the *Enable CDC* and *Enable automatic data transfer* check boxes.

- The *Enable CDC* option is available only when you create a new datastore. After you save a datastore, or when editing a datastore, the software disables the *Enable CDC* check box. Note that although a database datastore may have multiple configurations of different database types, if you enable CDC for a datastore then all configurations must use the same database type.
- The *Enable automatic data transfer* check box is selected by default when you create a new datastore and you chose `Database` for *Datastore type*. This check box is available when you edit an existing datastore. This check box displays for all databases except Attunity Connector, Memory, and Persistent Cache.

  > i Note
  >
  > The Enable automatic data transfer check box is not available for application datastores such as SAP and Oracle Applications.

  Keep Enable automatic data transfer selected to enable transfer tables in this datastore that the Data_Transfer transform can use to push down subsequent database operations.
- The *Force UTF-16 codepage* check box is available only when you select Persistent Cache as the database type. Selecting this option caches the data in a Unicode multi-byte format. Deselecting this option runs the data flow based on the engine runtime locale. You may want to use this option if you have several data flows that are dependent on each other and also have a mix of single-and multi-byte data. Setting this option ensures that the dependent data flows are compatible. However, if you have single-byte data and you select this option, you may see performance degradation in the data flow that uses this persistent cache.

Click *Advanced* to expand the datastore editor. The expanded window displays a grid of additional datastore options.

Creating a new Datastore—Advanced Configuration Options

You can toggle the *Advanced* button to hide and show the grid of additional datastore editor options.

The grid displays datastore configurations as column headings and lists datastore options in the left column. Each row represents a configuration option. Different options appear depending upon datastore type and (if applicable) database type and version. Specific options appear under group headings such as *Connection, General*, and *Locale*.

To improve readability, you can expand and collapse the datastore option groups. Each cell in the grid represents the value for a configuration option. If the value for a cell comes from a closed set, the cell becomes a drop-down list when you click it. If you are required to manually enter the value for an option, the cell becomes an text box when you click it.

If the *Database type* supports multiple configurations, the window also enables the *Edit...* button.

The Configurations Editor

Click the *Edit...* button to open the configurations editor which contains the grid of configuration options as well as an editing toolbar. The configurations editor is a subset of the datastore editor functionality. Use the configurations editor to add, edit, and remove datastore configurations.

The configurations editor always contains at least one configuration (initially that configuration reflects the first values set for the datastore). This first configuration is the default. When a datastore contains only one configuration (the default), you cannot remove it from the datastore. All subsequent configurations appear as additional columns in the grid.

The configurations editor provides a tool bar which includes commands to add, edit, and remove configurations.

From left to right, the toolbar buttons are:

Button	Button name	Description
	Create New Configuration	Adds a new configuration with no values.
	Duplicate Configuration	Creates a new configuration with identical settings as the selected configuration. The new configuration name must be unique, so the software uses the following naming convention: OldConfigurationName _Copy_CopyNumber. For example, if you duplicate a configuration called TestConfiguration, the software would name the duplicate TestConfiguration_Copy_1. If you do not rename the original or duplicate configuration and duplicate the original configuration again, the copy number appends by 1. So, the software would name the second duplicate TestConfiguration_Copy_2, and so forth.
	Rename Configuration	Shifts input focus to the name of the selected configuration so you can edit it.
	Delete Configuration	Removes the configuration from the datastore and its column from the grid.
AZ	Sort Configurations (Ascending)	Arranges the configurations by their names in ascending order. The arrangement is sensitive to the computer's system locale.
ZA	Sort Configurations (Descending)	Arranges the configurations by their names in descending order. The arrangement is sensitive to the computer's system locale.
	Move Default to First	Moves the default configuration to the first column in the list. Does not change the order of other columns.
	Create New Alias	Adds a new alias name for the datastore. To map individual configurations to an alias, enter the real owner name of the configuration in the grid.

Button	Button name	Description
	Delete Alias	Removes the selected alias name for the datastore.
	Expand All Categories	Opens all the nodes so that every configuration property is visible.
	Collapse All Categories	Closes all the nodes so that every configuration property is hidden.
	Show/Hide Details	This is a toggle to show additional datastore options on the dialog box: Database type, Number of Configurations, and CDC status.
Go to: Oracle1 ▾	Navigation box	This list contains the names of all configurations. Selecting a name from this list will (if necessary) scroll the configuration into view and highlight the configuration name in the grid.

These commands (except for the Navigation box) also appear on a shortcut menu when you right-click any active cell on the grid.

To save a newly-defined configuration and keep working in the configurations editor, click *Apply*. To save configuration changes and exit the configurations editor, click *OK*. Your configurations are saved in the same sequence shown in the configurations editor. To exit the configurations editor without saving changes, click *Cancel*.

You can also manage configurations by directly manipulating the grid.

- When a datastore contains more than one configuration, you can rearrange the order of configuration columns by clicking a configuration name and dragging it left or right.
- Double-click a configuration name to edit it.
- Right-click a configuration name or any active cell on the grid to select any of the following options from the shortcut menu:
  - *Create New Configuration* (see toolbar description)
  - *Duplicate Configuration* (see toolbar description)
  - *Rename Configuration* (see toolbar description)
  - *Delete Configuration* (see toolbar description)
  - *Sort Configurations in Ascending Order* (see toolbar description)
  - *Sort Configurations in Descending Order* (see toolbar description)
  - *Move Default Configuration to First Column* (see toolbar description)
  - *Expand All Categories* (see toolbar description)
  - *Collapse All Categories* (see toolbar description)
  - *Add Linked Datastore*
  - *Delete Linked Datastore*
  - *Create New Alias*
  - *Delete Alias*

Using multiple configurations with database datastores can minimize your efforts to port existing jobs from one database type and version to another. The datastore editor supports quick creation of multiple configurations by allowing you to duplicate and rename configurations. Duplicating a configuration copies its options to create another configuration.

Because each datastore must have only one Default configuration (used to browse, search, and import metadata), when you select *Yes* as the default for any one configuration, the grid automatically sets the Default configuration value for the others to *No*.

> Note
>
> While you can change the Default configuration value from *No* to *Yes*, you cannot change the value from *Yes* to *No*. If you attempt to do so, the Designer displays an error message instructing you to select *Yes* for another configuration instead.

Adding New Configurations

When you add new configurations, the software modifies the language of data flows in the datastore if the data flows contain any of the following objects:

- Table targets
- Table transfer type used in Data_Transfer transform as a target
- SQL transforms

The software adds the target options and SQL transform text to additional datastore configurations based their definitions in an existing configuration.

This functionality operates in the following ways:

- If a new configuration has the same database type and the same or newer version as an old configuration, then the software automatically uses the existing SQL transform, target table editor, and Data_Transfer transform editor values (including bulk loader options).
- If the database type and version are not already associated with (or if the version is older than) any existing configuration, you can use the values from an existing database type and version by selecting that option from the *Use values from* list.
  The Use values from list always contains the following options:
  - Default values
  - Database type and version for each configuration currently associated with the datastore

  So if your datastore contains two configurations, for example one for Oracle 9i and one for Microsoft SQL Server 2000, when you create a third configuration, (in this example, for DB2) you will see Default values, Oracle 9i and Microsoft SQL Server 2000 as the options in the Use values from list.
  Default values are the same defaults that appear for all database targets, Data_Transfer target tables, and SQL transforms. Default SQL text is always blank. Some target option default values are:
  Row commit size = 1000
  Column comparison = Compare by name
  Delete data from table before loading = not selected
  Drop and re-create table = not selected for regular tables (Selected for template tables)
- If you select the *Restore values if they already exist* check box (pre-selected as default), the software creates the new configuration then determines whether SQL transform, target table editor, or Data_Transfer transform editor values already exist for the new database. If the database values already exist, the software

restores the bulk load option. However, if no values exist for the database, the software sets the bulk load option to *None*, the default value.
Also, if you deselect Restore values if they already exist, the software sets the bulk load option to None, the default value.

Example: Suppose you are working in a multi-user environment and have a local datastore with configurations for Oracle 9i and SQL Server 2000. You also have existing data flows that use target tables, Data_Transfer target tables, or SQL transforms from this datastore. You then delete Oracle 9i (perhaps because you checked out a different version of the datastore from the central repository). Later, you want to add an Oracle 9i configuration to this datastore.

Deleting a version causes the software to remove the configuration, but not the target table, Data_Transfer target table, and SQL transform values. If you select *Restore values if they already exist* when you create a new configuration, the software determines whether values already exist for the database. If the software cannot find these values, the Designer uses values specified in the *Use values from* box.

After you click *Apply* to save a new configuration, the software:

- Copies any existing SQL transform, target table editor, and Data_Transfer target table editor values, and
- Displays a report of the modified objects in a popup window as well as in the Designer Output window.

The report shows the following information:

Report column	Description
data flow	Names of the data flows where language was modified
Modified Object	Objects in the data flows that were affected
Object Type	Types of the objects affected (table target or SQL transform)
Usage	Usage of the objects (source or target)
Has Bulk Loader	Whether the objects have a bulk loader
Bulk Loader Copied	Whether the bulk loader option was copied
Values Existed	Whether there were previous values
Values Restored	Whether the previous values were restored

You can use this report as a guide to manually change the values for options of targets, Data_Transfer target tables, and SQL transforms, as needed. In the pop-up window, you can sort results by clicking on column headers. You can also save the output to a file. The popup appears after each newly-added configuration.

SAP Data Services also clears and displays the results in the Output window after each newly-added configuration. Because the datastore editor windows are modal, you cannot see the entire Output window or manipulate it. However, you can double-click one of the objects on the report and to view the data flow.

Configurations with Different Database Types

When a datastore contains multiple configurations of different database types, the rows show the options for all configurations.

When an option does not apply to a configuration, the cell displays N/A in gray and does not accept input. Cells that correspond to a group header such as Connection and Locale display hashed gray lines and also do not accept input.

Importing database links

Use this datastore option to import and configure a database link in the Designer.

Related Information

Performance Optimization Guide: Maximizing Push-Down Operations, DataTransfer transform for push-down operations [page 2122]
Working with Aliases [page 868]
Designer Guide: Datastores, Adapter datastores [page 233]
Designer Guide: Datastores, Linked datastores [page 231]

3.2.2.7.1.1 To link a target datastore to a source datastore using a database link

1. From the *Datastores* tab in the object library, right-click a target datastore and select *Edit*.

   If the database type supports database links, the list of configuration options includes the *Linked Datastores* option, in the *Advanced* section.

   > i Note
   >
   > The datastore editor allows you to edit database links on target datastores for the default configuration only. So, if your target datastore contains multiple configurations (for example: Config1, Config2, and Config3), change your default configuration before you attempt to import or edit links that apply to it (for example, make Config2 the default if you want to edit it).

2. Click the *Linked Datastores* label.

   The *Add Linked Datastore* window opens.
3. From the *Add Linked Datastore* window, select a datastore that your target datastore will be linked to based on the settings in the database link you want to import.

   For example if your target datastore is DS_Emp (employee information) and the database link you want to import will associate employee information with sales information, select DS_Sales (sales information).

   The datastores in the list box have database types supported for linked datastores.

   > i Note
   >
   > The datastore editor allows only one database link between a target datastore and a source datastore pair. Therefore, if target datastore B already has a link to source datastore A, you cannot import another database link that associates datastore B with datastore A.

4. Click *OK*.

   The *Datastore Editor* window displays the datastore that you selected.

5. Select the list button to the right of `Not Linked` or double-click the cell.

   The *Database Link* window opens.
6. To link to a datastore or to change the existing link, select *Use the database link*.

   > **Note**
   >
   > To remove an existing link, select Do not link.
7. Select a database link from the list read from the default configuration connection of the target datastore you are editing.

   This list box contains links that you previously defined on the DBMS.
8. Select the source datastore configuration that you want to use with this database link.
9. (Optional) Select *Details* to view additional information about the links or to test them.

   The check mark indicates the link to use. If you use the *Details* window, click OK when you are finished.
10. From the *Database Link* dialog, click *OK*.

3.2.2.7.1.2 Working with Aliases

Use this option to define aliases for your datastore. After you create an alias (for example, ALIAS1, ALIAS2), navigate horizontally to each configuration and define the owner name to which that alias name maps.

Note that SAP Data Services does not label alias owner names in the configurations grid.

When you delete an alias name, the delete operation applies to the entire datastore (all configurations). the software removes the selected row which includes the alias and all assigned owner names.

3.2.2.7.2 Database datastores

You can define datastores so that SAP Data Services can read from and write to the following types of databases:

- Attunity Connector (use for mainframe systems)
- Data Federator (read only)
- DB2
- HP Neoview
- Informix
- Memory
- Microsoft SQL Server
- MySQL
- Netezza
- ODBC
- Oracle
- Persistent Cache
- SAP HANA (ODBC)
- SQL Anywhere

- SAP SAP Sybase ASE
- SAP SAP Sybase IQ
- Teradata

Each database requires its own connection information in the datastore definition.

For a description of the datastore connection information and options specific to each database, see the tables in this section.

i Note

The *Enable CDC* option is available with a subset of the databases. When the *Enable CDC* option is checked, the following Advanced option groups do not display because a CDC datastore is read-only and you can only use it as a source: *General*, *Bulk Loader*, and *FTP*.

Related Information

ODBC [page 885]
Designer Guide: What are datastores [page 209]
Performance Optimization Guide: Using Bulk Loading [page 2164]

3.2.2.7.2.1 Attunity

Table 19: Main window

Attunity option	Possible values	Description
Data source	Refer to the requirements of your database	Type the Attunity data source name(s) as defined in Attunity Studio. Separate multiple data source names with semicolons.
Host location	Computer name, fully qualified domain name, or IP address	Type the name of the Attunity server computer (host).
Port	Positive integer	Type the port number for the Attunity server.
Attunity workspace	Refer to the requirements of your database	Type the workspace name under which the data sources are defined in Attunity Studio.
User name	Alphanumeric characters and underscores	Type the user name of the account through which SAP Data Services accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Type the user's password.
Enable CDC	-	Select to enable changed data capture for this datastore.

Table 20: General (these options do not appear for CDC datastores)

Attunity option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.

Table 21: Locale

Attunity option	Possible values	Description
Language	-	See the section "Locales and Multi-Byte Functionality."
Code page	-	See the section "Locales and Multi-Byte Functionality."
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 22: Session

Attunity option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon	Additional session parameters specified as valid SQL statement(s)

Table 23: Aliases

Attunity option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

3.2.2.7.2.2 Data Federator

Any decimal column imported to SAP Data Services from an SAP Data Federator data source is converted to the decimal precision and scale(28,6).

Any varchar column imported to the software from a SAP Data Federator data source is varchar(1024).

You may change the decimal precision or scale and varchar size within the software after importing form the SAP Data Federator data source.

3.2.2.7.2.3 DB2

Table 24: Main window

DB2 option	Possible values	Description
Database version	DB2 UDB `<version number>`	Select the version of your DB2 client. This is the version of DB2 that this datastore accesses.
Use data source name (DSN)	Check box selected or not selected	Select to use DSN to connect to the database. By default, this option is not selected and a server name (also known as DSN-less) connection will be used. For a DSN-less connection, you must fill in *Database server name*, *Database name*, and *Port*. If you select this checkbox, you must fill in *Data source name*
Database server name	Refer to the requirements of your database	Type the DB2 database server name. This option is required if you did not select *Use data source name (DSN)*.
Database name	Refer to the requirements of your database	Type the name of the database defined in DB2. This option is required if you did not select *Use data source name (DSN)*.
Port	Five digit integer Default: 50000	Enter the number of the database port. This option is required if you did not select *Use data source name (DSN)*.
Data source name	Refer to the requirements of your database	Type the data source name defined in DB2 for connecting to your database. This option is required when you select *Use data source name (DSN)*. If you are going to use the Auto correct load feature for DB2 targets, be sure that your data source allows your user name to create or replace stored procedures.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which SAP Data Services accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.

Table 25: General

DB2 option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this da-

DB2 option	Possible values	Description
		tastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click *Browse*	Enter the location where command and data files are written for bulk loading. For Solaris systems, the path name must be less than 80 characters. You can enter a variable for this option.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.

Table 26: Locale

DB2 option	Possible values	Description
Language	-	See the section "Locales and Multi-Byte Functionality."
Code page	-	See the section "Locales and Multi-Byte Functionality."
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 27: Bulk loader

DB2 option	Possible values	Description
Bulk loader user name	Alphanumeric characters and underscores or blank	The name used when loading data with the bulk loader option. For bulk loading, you might specify a different user name. For example, specify a user who has import and load permissions.
Bulk loader password	Alphanumeric characters, underscores, and punctuation, or blank	The password used when loading with the bulk loader option.
DB2 server working directory	Directory path or click *Browse*	The working directory for the load utility on the computer that runs the DB2 server. You must complete this field whenever the DB2 server and the Job Server run on separate machines.

Table 28: FTP

DB2 option	Possible values	Description
FTP host name	Computer name, fully qualified domain name, or IP address	If this field is left blank or contains the name of the computer (host) where the Job Server resides, the software assumes that DB2 and and the software share the same computer and that FTP is unnecessary. When FTP is unnecessary, all other FTP-related fields can remain blank.

DB2 option	Possible values	Description
FTP login user name	Alphanumeric characters and underscores, or blank	Must be defined to use FTP.
FTP login password	Alphanumeric characters, underscores, and punctuation, or blank	Must be defined to use FTP.
Session		
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon.	Additional session parameters specified as valid SQL statement(s)
Aliases (Click here to create)		
Aliases	-	Enter the alias name and the owner name to which the alias name maps.
Linked Datastores (Click here to create)		
Datastore name	Alphanumeric characters and underscores or blank	The name of a datastore to which you linked the current datastore configuration in preparation to import a database link

3.2.2.7.2.4 HP Neoview

Table 29: Main window

HP Neoview option	Possible values	Description
Database version	Currently supported versions `<version number>`	Select the version of your HP Neoview client. This is the version of HP Neoview that this datastore accesses.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which the software accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.
Enable automatic data transfer		The Enable automatic data transfer check box is selected by default when you create a new datastore and you chose Database for Datastore type. Keep Enable automatic data transfer selected to enable transfer tables in this datastore that the Data_Transfer transform can use to push down subsequent database operations.

Table 30: General

HP Neoview option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click *Browse*	Enter the location where command and data files are written for bulk loading. For Solaris systems, the path name must be less than 80 characters. You can enter a variable for this option.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.

Table 31: Neoview Transporter

HP Neoview option	Possible values	Description
Data source	Alphanumeric characters and underscores	Type the data source name of the ODBC and JDBC connection to the HP Neoview system. The name must be unique.
System	Alphanumeric characters and underscores	Type the name of the primary segment, for example, the HP Neoview system named neo0101.domain.com has the system name of neo0101.
Access point URL	Alphanumeric characters and underscores	Enter the URL for internal database connections. For example, the HP Neoview system named neo0101.domain.com would have a JDBC connection URL as `jdbc:hpt4jdbc://eno0101.domain.com:18650`.
User name	Alphanumeric characters and underscores	Type the username for connecting to the data source.
Password	Alphanumeric characters, underscores, and punctuation	Type the password for connecting to the data source.

Table 32: Session

HP Neoview option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon	Additional session parameters specified as valid SQL statement(s).

Table 33: Aliases (Click here to create)

HP Neoview option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

3.2.2.7.2.5 Informix

Table 34: Main window

Informix option	Possible values	Description
Database version	Informix IDS `<version number>`	Select the version of your Informix client. This is the version of Informix that this datastore accesses.
Use data source name (DSN)	Check box selected or not selected	Select to use DSN to connect to the database. By default, this option is not selected and a server name (also known as DSN-less) connection will be used. For a DSN-less connection, you must fill in *Database server name*, *Database name*, and *Port*. If you select this checkbox, you must fill in *Data source name*.
Database server name	Refer to the requirements of your database	Type the Informix database server name. This name is the Informix server instance name, not the host name. This option is required if you did not select *Use data source name (DSN)*.
Database name	Refer to the requirements of your database	Type the name of the database defined in Informix . This option is required if you did not select *Use data source name (DSN)*.
Port	Four digit integer Default: 1526	Type the port number to connect to this database. This option is required if you did not select *Use data source name (DSN)*.
Data source name	Refer to the requirements of your database	Type the Data Source Name defined in the ODBC. This option is required when you select *Use data source name (DSN)*.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which SAP Data Services accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.

Table 35: General

Informix option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is

Informix option	Possible values	Description
		the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click *Browse*	Enter the directory where the software writes sql, control, command, and data files for bulk loading. For Solaris systems, the path name must be less than 80 characters. You can enter a variable for this option.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore.

Table 36: Locale

Informix option	Possible values	Description
Language	-	See the section "Locales and Multi-Byte Functionality."
Code page	-	See the section "Locales and Multi-Byte Functionality."
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 37: Session

Informix option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon.	Additional session parameters specified as valid SQL statement(s).
Aliases (Click here to create)		
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

i Note

To use large object types with Informix, you must first enable large object support within the Informix ODBC driver options.

For Windows, in the *Informix ODBC Driver Setup* screen used for configuration of the Informix ODBC driver, check *Report Standard ODBC Types Only* and click *OK* to save the change.

For UNIX platforms, edit the `odbc.ini` file to set the option `NEEDODBCTYPESONLY=1`.

For more information about Informix ODBC driver configuration, see your Informix ODBC driver configuration documentation.

3.2.2.7.2.6 Memory

Table 38: Bulk Loader

Memory option	Possible values	Description
JS and DB on same machine	Yes, No	Indicate whether the Job Server and database are on the same computer.

Related Information

Locales and Multi-byte Functionality [page 1793]

3.2.2.7.2.7 Microsoft SQL Server

i Note

If there is a problem when processing Microsoft SQL Server data using a UNIX Job Server, Data Services will issue an error message that might contain a DataDirect internal error message. DataDirect's error-message syntax resembles the following structure: `<vendor-ODBCcomponent: message>`

For more information about message text or codes that DataDirect issues, consult DataDirect documentation or web site.

Table 39: Main window

Microsoft SQL Server option	Possible values	Description
Database version	Microsoft SQL Server `<version number>`	Select the version of your SQL Server client. This is the version of SQL Server that this datastore accesses.
Database server name	Computer name, fully qualified domain name, or IP address	Enter the name of machine where the SQL Server instance is located.
Database name	Refer to the requirements of your database	Enter the name of the database to which the datastore connects.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which Data Services accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.
Enable CDC	-	Select to enable changed data capture for this datastore.

Microsoft SQL Server option	Possible values	Description
Connection		
Use Windows Authentication	No, Yes	Select whether to use Windows authentication or Microsoft SQL Server authentication to connect to this datastore. Defaults to No. For more information on how to use Windows authentication with Microsoft SQL Server, refer to the Microsoft SQL Server documentation. i Note Windows authentication is not supported on UNIX.

Table 40: General

Microsoft SQL Server option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. You can enter a variable for this option.

Table 41: Locale

Microsoft SQL Server option	Possible values	Description
Language	-	See the section "Locales and Multi-Byte Functionality."
Code page	-	See the section "Locales and Multi-Byte Functionality."
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 42: Session

Microsoft SQL Server option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon	Additional session parameters specified as valid SQL statement(s)

Table 43: Aliases (Click here to create)

Microsoft SQL Server option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

Table 44: Linked Datastores (Click here to create)

Microsoft SQL Server option	Possible values	Description
Datastore Name	Alphanumeric characters and underscores or blank	The name of a datastore to which you linked the current datastore configuration in preparation to import a database link

3.2.2.7.2.8 MySQL

Table 45: Main window

MySQL option	Possible values	Description
Database version	Currently supported versions `<version number>`	Select the version of your MySQL client. This is the version of MySQL that this datastore accesses.
Use data source name (DSN)	Check box selected or not selected	Select to use DSN to connect to the database. By default, this option is not selected and a server name (also known as DSN-less) connection will be used. For a DSN-less connection, you must fill in *Database server name*, *Database name*, and *Port*. If you select this checkbox, you must fill in *Data source name*
Database server name	Refer to the requirements of your database	Type the MySQL database server name. This option is required if you did not select *Use data source name (DSN)*.
Database name	Refer to the requirements of your database	Type the name of the database defined in MySQL. This option is required if you did not select *Use data source name (DSN)*.
Port	Refer to the requirements of your database	Enter the number of the database port. This option is required if you did not select *Use data source name (DSN)*.
Data source name	Refer to the requirements of your database	Type the data source name defined in MySQL for connecting to your database. This option is required when you select *Use data source name (DSN)*. If you are going to use the Auto correct load feature for MySQL targets, be sure that your data source allows your user name to create or replace stored procedures.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which SAP Data Services accesses the database.

MySQL option	Possible values	Description
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.
Enable automatic data transfer		The Enable automatic data transfer check box is selected by default when you create a new datastore and you chose Database for Datastore type. Keep *Enable automatic data transfer* selected to enable transfer tables in this datastore that the Data_Transfer transform can use to push down subsequent database operations.

Table 46: Connection

MySQL option	Possible values	Description
Additional connection parameters	Alphanumeric characters and underscores, or blank	Enter information for any additional parameters that the data source supports (parameters that the data source's ODBC driver and database support). Use the format: `<parameter1=value1; parameter2=value2>`

Table 47: General

MySQL option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click *Browse*	Enter the location where command and data files are written for bulk loading. For Solaris systems, the path name must be less than 80 characters. You can enter a variable for this option.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.
Database Server working directory	Directory path or click *Browse*	The working directory for the load utility on the computer that runs the MySQL server. You must complete this field whenever the MySQL server and the Job Server run on separate machines.

Table 48: Locale

MySQL option	Possible values	Description
Language	-	See the section "Locales and Multi-Byte Functionality."

MySQL option	Possible values	Description
Code page	-	See the section "Locales and Multi-Byte Functionality."
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 49: FTP

MySQL option	Possible values	Description
FTP host name	Computer name, fully qualified domain name, or IP address	If this field is left blank or contains the name of the computer (host) where the Job Server resides, the software assumes that MySQL and and the software share the same computer and that FTP is unnecessary. When FTP is unnecessary, all other FTP-related fields can remain blank.
FTP login user name	Alphanumeric characters and underscores, or blank	Must be defined to use FTP.
FTP login password	Alphanumeric characters, underscores, and punctuation, or blank	Must be defined to use FTP.
FTP host working directory	Absolute file path Virtual file path (Windows servers only)	The location on the database server where Data Services transfers the data file between the Job Server and the MySQL server. For Windows servers only, you can configure a path to a virtual directory.

Table 50: ODBC Miscellaneous

MySQL option	Values	Description
Date format	`yyyy.mm.dd` or other combinations	Enter a date format supported by the data source (a date format that the data source's ODBC driver and database supports).
Time format	`hh24:mi:ss` or other combinations	Enter a time format supported by the data source (a time format that the data source's ODBC driver and database supports).
Date-time format	`yyyy.mm.dd hh24:mi:ss` or other combinations	Enter a date-time format supported by the data source (a date-time format supported by the data source's ODBC driver and database).
Decimal separator	.(peroid) or , (comma)	Enter the character that the data source uses to separate the decimal portion of a number.
Data type conversion support	Automatic, ODBC syntax, No, SQL-92 syntax	When there's a data type mismatch in an expression, the software automatically generates an explicit convert function call.
NVL support	Automatic, ODBC syntax, No, custom	If the input value is NULL, replace with the specified value.

MySQL option	Values	Description
Ifthenelse support	Yes, No	Allows conditional logic in mapping and selection operations.

Table 51: Session

MySQL option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon	Additional session parameters specified as valid SQL statement(s).

Table 52: Aliases (Click here to create)

MySQL option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

3.2.2.7.2.9 Netezza

Table 53: Main window

Netezza option	Possible values	Description
Database version	Currently supported versions `<version number>`	Select the version of your Netezza client. This is the version of Netezza that this datastore accesses.
Use data source name (DSN)	Check box selected or not selected	Select to use DSN to connect to the database. By default, this option is not selected and a server name (also known as DSN-less) connection will be used. For a DSN-less connection, you must fill in *Database server name*, *Database name*, and *Port*. If you select this checkbox, you must fill in *Data source name*
Database server name	Refer to the requirements of your database	Type the Netezza database server name. This option is required if you did not select *Use data source name (DSN)*.
Database name	Refer to the requirements of your database	Type the name of the database defined in Netezza. This option is required if you did not select *Use data source name (DSN)*.
Port	Refer to the requirements of your database	Enter the number of the database port. This option is required if you did not select *Use data source name (DSN)*.

Netezza option	Possible values	Description
Data source name	Refer to the requirements of your database	Type the data source name defined in Netezza for connecting to your database. This option is required when you select *Use data source name (DSN)*. If you are going to use the Auto correct load feature for Netezza targets, be sure that your data source allows your user name to create or replace stored procedures.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which SAP Data Services accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.
Enable automatic data transfer		The Enable automatic data transfer check box is selected by default when you create a new datastore and you chose Database for Datastore type. Keep *Enable automatic data transfer* selected to enable transfer tables in this datastore that the Data_Transfer transform can use to push down subsequent database operations.

Table 54: Connection

Netezza option	Possible values	Description
Additional connection parameters	Alphanumeric characters and underscores, or blank	Enter information for any additional parameters that the data source supports (parameters that the data source's ODBC driver and database support). Use the format: `<parameter1=value1; parameter2=value2>`

Table 55: General

Netezza option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click *Browse*	Enter the location where command and data files are written for bulk loading. For Solaris systems, the path name must be less than 80 characters. You can enter a variable for this option.

Netezza option	Possible values	Description
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.
Database Server working directory	Directory path or click *Browse*	The working directory for the load utility on the computer that runs the Netezza server. You must complete this field whenever the Netezza server and the Job Server run on separate machines.

Table 56: Locale

Netezza option	Possible values	Description
Language	-	See the section "Locales and Multi-Byte Functionality."
Code page	-	See the section "Locales and Multi-Byte Functionality."
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 57: FTP

Netezza option	Possible values	Description
FTP host name	Computer name, fully qualified domain name, or IP address	If this field is left blank or contains the name of the computer (host) where the Job Server resides, the software assumes that Netezza and and the software share the same computer and that FTP is unnecessary. When FTP is unnecessary, all other FTP-related fields can remain blank.
FTP login user name	Alphanumeric characters and underscores, or blank	Must be defined to use FTP.
FTP login password	Alphanumeric characters, underscores, and punctuation, or blank	Must be defined to use FTP.
FTP host working directory	Absolute file path Virtual file path (Windows servers only)	The location on the database server where Data Services transfers the data file between the Job Server and the Netezza server. For Windows servers only, you can configure a path to a virtual directory.

Table 58: ODBC Miscellaneous

Netezza option	Possible values	Description
Date format	`yyyy.mm.dd` or other combinations	Enter a date format supported by the data source (a date format that the data source's ODBC driver and database supports).

Netezza option	Possible values	Description
Time format	`hh24:mi:ss` or other combinations	Enter a time format supported by the data source (a time format that the data source's ODBC driver and database supports).
Date-time format	`yyyy.mm.dd hh24:mi:ss` or other combinations	Enter a date-time format supported by the data source (a date-time format supported by the data source's ODBC driver and database).
Decimal separator	.(peroid) or , (comma)	Enter the character that the data source uses to separate the decimal portion of a number.
Data type conversion support	Automatic, ODBC syntax, No, SQL-92 syntax	When there's a data type mismatch in an expression, the software automatically generates an explicit convert function call.
NVL support	Automatic, ODBC syntax, No, custom	If the input value is NULL, replace with the specified value.
Ifthenelse support	Yes, No	Allows conditional logic in mapping and selection operations.

Table 59: Session

Netezza option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon	Additional session parameters specified as valid SQL statement(s).

Table 60: Aliases (Click here to create)

Netezza option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

3.2.2.7.2.10 ODBC

To define an ODBC datastore connection, you need to define a data source, a user name, a password if applicable, and optionally a set of advanced options.

Selecting an ODBC data source

You can select a data source in one of the following ways. In the *Data source* field of the ODBC datastore editor:

- From the drop-down list, click an existing data source
- Type the name of a data source
- Click *ODBC Admin* to launch the Windows ODBC Data Source Administrator where you create or configure data sources. After closing the ODBC Data Source Administrator, you can select a newly created data source from the datastore editor's drop-down list.

Defining ODBC datastore options

To define options for an ODBC datastore, click *Advanced*. For each option to configure, you can select a value from its drop-down list, or many options allow you to type a custom value.

Most ODBC datastore options include the following values.

Automatic

When you create a new ODBC datastore, most options default to *Automatic*. With this setting, if you do not know if the ODBC driver supports an option, SAP Data Services queries the driver to determine its capabilities. If the driver supports that option, the software pushes down the operation to the ODBC database. If the ODBC driver does not support that option, the software executes the operation internally.

To circumvent possible inconsistencies with the ODBC driver, you might need to specify an option other than *Automatic*. If you select anything other than *Automatic*, the software does not query the driver for that particular capability. Most options in the ODBC datastore editor provide some or all of the following choices.

ODBC syntax

The software assumes the ODBC driver supports the function/capability and uses ODBC syntax.

For example, for the ABSOLUTE function, the syntax is:

```
{fn abs (TAB1.COL1)}
```

SQL-92

The software assumes the ODBC driver supports the function/capability and uses SQL-92 syntax.

For example, when the software generates an explicit CONVERT function, the syntax is:

```
CAST (TAB1.VC_COL AS SQL_INTEGER)
```

No

The software assumes the ODBC driver does not support the function/capability and executes it internally.

Custom

Many functions allow you to type in the specific function call to use for that option. The software assumes the ODBC driver supports the function/capability.

> **Note**
>
> You cannot specify the signature of the function; it will be the same as in the ODBC signature.

For example, for the string function *Upper case*, instead of using `{fn ucase(...)}`, you can type in the *Upper case* option field `upper`. The software generates:

```
upper(TAB1.VC_COL)
```

The following tables describes the fields and options in the ODBC datastore editor.

Table 61: Main window

ODBC option	Values	Description
Data source	Refer to the requirements of your database.	Select or type the Data Source Name defined in the ODBC Administrator for connecting to your database.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which the software accesses the database. **Note** If you use the Neoview utility Nvtencrsrv for storing the encrypted words in the security file when using Neoview Transporter, enter the encrypted user name
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password. **Note** If you use the Neoview utility Nvtencrsrv for storing the encrypted words in the security file when using Neoview Transporter, enter the encrypted password.
ODBC Admin button		Click to launch the Windows ODBC Data Source Administrator where you create or configure data sources. After closing the ODBC Data Source Administrator, you can select a newly created data source from the datastore editor's drop-down list.

Table 62: Connection

ODBC option	Values	Description
Additional connection information	Alphanumeric characters and underscores, or blank	Enter information for any additional parameters that the data source supports (parameters that the data source's ODBC driver and database support). Use the format: `<parameter1=value1; parameter2=value2>`

Table 63: General

ODBC option	Values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click *Browse*	Enter the directory where the software writes sql, control, command, and data files for bulk loading. For Solaris systems, the path name must be less than 80 characters. You can enter a variable for this option.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. You can enter a variable for this option.
Database server working directory	Directory path or click *Browse*	A working directory on the database server that stores files such as logs. Must be defined to use FTP.

Table 64: Locale

ODBC option	Values	Description
Language		See the section "Locales and Multi-Byte Functionality"
Code page		See the section "Locales and Multi-Byte Functionality"
Server code page		See the section "Locales and Multi-Byte Functionality"

Table 65: Neoview Transporter

ODBC option	Values	Description
Data source	Alphanumeric characters and underscores	Type the data source name of the ODBC and JDBC connection to the HP Neoview system. The name must be unique.
System	Alphanumeric characters and underscores	Type the name of the primary segment, for example, the HP Neoview system named neo0101.domain.com has the system name of **`neo0101`**.
Access point URL	Alphanumeric characters and underscores	Enter the URL for internal database connections. For example, the HP Neoview system named neo0101.domain.com would have a JDBC connection URL as `jdbc:hpt4jdbc://eno0101.domain.com:18650`.
User name	Alphanumeric characters and underscores	Type the username for connecting to the data source.
Password	Alphanumeric characters, underscores, and punctuation	Type the password for connecting to the data source.

Table 66: FTP

Note

If this datastore is not being used specifically for Netezza bulk loading, the software ignores any FTP option entries.

ODBC option	Values	Description
FTP host name	Computer name, fully qualified domain name, or IP address	For a Netezza server, type the name of the Netezza server computer (host). Must be defined to use FTP.
FTP login user name	Alphanumeric characters and underscores, or blank	Must be defined to use FTP.
FTP login password	Alphanumeric characters, underscores, and punctuation, or blank	Must be defined to use FTP.
FTP host working directory	Absolute file path	The location on the database server from where the software retrieves diagnostic files generated by the database's bulk loader. It must be accessible from the FTP server. It is usually the same as the database's working directory. If unsure, contact your system administrator. Note Configure the FTP server to accept an absolute path.

Table 67: ODBC Capability Support

ODBC option	Values	Description
Array fetch	Automatic, No	If you encounter errors when reading from an ODBC datastore, especially if the error message involves the ODBC call SQLFetchScroll, it is safe to assume that your ODBC driver does not support array fetch. In this case, select the *No* value to turn off the array fetch capability. The software fetches one row at a time from the ODBC data source. The *No* value causes all Source Table Editors and SQL Transform Editors that use this ODBC datastore to not display the *Array fetch size* performance option.
Parameterized SQL	Automatic, No	By using parameterized SQL, the software generates SQL statements with parameters instead of literal values, which can significantly improve performance.
Outer join	Automatic, ODBC syntax, SQL-92 syntax, No	Determines whether the ODBC driver supports outer join syntax.
Auto commit	Automatic, Yes, No	Determines whether the ODBC driver supports auto commit.
Server sorts in binary	Yes, No	Determines whether the server performs binary sort.

Table 68: ODBC Math Function Support

ODBC option	Values	Description
Absolute	Automatic, ODBC syntax, No, custom	Returns the absolute value of an input number.
Ceiling	Automatic, ODBC syntax, No, custom	Returns the smallest integer value greater than or equal to an input number.
Floor	Automatic, ODBC syntax, No, custom	Returns the largest integer value less than or equal to an input number.
Round	Automatic, ODBC syntax, No, custom	Rounds a given number to the specified precision.
Truncate	Automatic, ODBC syntax, No, custom	Truncates a given number to the specified precision.
Sqrt	Automatic, ODBC syntax, No, custom	Returns the square root of the input number.
Log	Automatic, ODBC syntax, No, custom	Returns the base-10 logarithm of the given numeric expression.
Ln	Automatic, ODBC syntax, No, custom	Returns the natural logarithm of the given numeric expression.
Power	Automatic, ODBC syntax, No, custom	Returns the value of the given expression to the specified power.
Mod	Automatic, ODBC syntax, No, custom	Returns the remainder when one number is divided by another.

Table 69: ODBC String Function Support

ODBC option	Values	Description
Lower case	Automatic, ODBC syntax, No, custom	Changes the characters in a string to lowercase.
Upper case	Automatic, ODBC syntax, No, custom	Changes the characters in a string to uppercase.
Rtrim blanks	Automatic, ODBC syntax, No, custom	Removes blank characters from the end of a string.
Ltrim blanks	Automatic, ODBC syntax, No, custom	Removes blank characters from the start of a string.
Length	Automatic, ODBC syntax, No, custom	Returns the number of characters in a given string.
Substring	Automatic, ODBC syntax, No, custom	Returns a specific portion of a string starting at a given point in the string.
Soundex	Automatic, ODBC syntax, No, custom	Returns the soundex encoding of the input string.

Table 70: ODBC Date Function Support

ODBC option	Values	Description
System date	Automatic, ODBC syntax, No, custom	Returns the current date as listed by the Job Server's operating system.
System time	Automatic, ODBC syntax, No, custom	Returns the current time as listed by the operating system.
Week	Automatic, ODBC syntax, No, custom	Determines the week in the year in which the given date falls.
Month	Automatic, ODBC syntax, No, custom	Determines the month in which the given date falls.
Quarter	Automatic, ODBC syntax, No, custom	Determines the quarter in which the given date falls.
Year	Automatic, ODBC syntax, No, custom	Determines the year in which the given date falls.
Day of month	Automatic, ODBC syntax, No, custom	Determines the day in the month on which the given date falls.
Day of year	Automatic, ODBC syntax, No, custom	Determines the day in the year on which the given date falls.

Table 71: ODBC Aggregate Function Support

ODBC option	Values	Description
Average	Automatic, SQL-92 syntax, No	Calculates the average of a given set of values.
Count	Automatic, SQL-92 syntax, No	Counts the number of values in a table column.
Count Distinct	Automatic, SQL-92 syntax, No	Counts the number of distinct non-NULL values in a table column.
Max	Automatic, SQL-92 syntax, No	Returns the maximum value from a list.
Min	Automatic, SQL-92 syntax, No	Returns the minimum value from a list.
Sum	Automatic, SQL-92 syntax, No	Calculates the sum of a given set of values.

Table 72: Miscellaneous

ODBC option	Values	Description
Date format	`yyyy.mm.dd` or other combinations	Enter a date format supported by the data source (a date format that the data source's ODBC driver and database supports).

ODBC option	Values	Description
Time format	`hh24:mi:ss` or other combinations	Enter a time format supported by the data source (a time format that the data source's ODBC driver and database supports).
Date-time format	`yyyy.mm.dd hh24:mi:ss` or other combinations	Enter a date-time format supported by the data source (a date-time format supported by the data source's ODBC driver and database).
Decimal separator	.(peroid) or , (comma)	Enter the character that the data source uses to separate the decimal portion of a number.
Data type conversion support	Automatic, ODBC syntax, No, SQL-92 syntax	When there's a data type mismatch in an expression, the software automatically generates an explicit convert function call.
NVL support	Automatic, ODBC syntax, No, custom	If the input value is NULL, replace with the specified value.
Ifthenelse support	Yes, No	Allows conditional logic in mapping and selection operations.
NVARCHAR type name	<Unknown> NVARCHAR NVARCHAR2	For loading multibyte data to template tables, select the option depending on the database type: *NVARCHAR*: All supported databases except Oracle *NVARCHAR2*: Oracle databases only In the template table target editor, setting the option *Use NVARCHAR for VARCHAR columns in supported databases* to Yes enables this data type conversion. See also *Template table* [page 996].

Table 73: Session

ODBC option	Values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by a semicolon	Additional session parameters specified as valid SQL statement.

Table 74: Aliases

ODBC option	Values	Description
Aliases		Enter the alias name and the owner name to which the alias name maps.

Related Information

Administrator Guide: Configuring ODBC data sources on UNIX [page 61]

3.2.2.7.2.11 Oracle

Table 75: Main window

Oracle option	Possible values	Description
Database version	Oracle `<version number>`	Select the version of your Oracle client. This is the version of Oracle that this datastore accesses.
Use TNS name	Check box selected or not selected	Select to use TNS to connect to the database. By default, this option is not selected and a server name (also known as TNS-less) connection will be used. For a TNS-less connection, you must fill in *Hostname*, *SID*, and *Port*. If you select this checkbox, you must fill in *TNS name*
Hostname	Computer name, fully qualified domain name, or IP address	Enter the name of machine where the Oracle Server instance is located. This option is required if you did not select *Use TNS name*.
SID	Refer to the requirements of your database	Enter the System ID for the Oracle database. This option is required if you did not select *Use TNS name*.
Port	Four digit integer Default: 1521	Enter the port number to connect to this Oracle Server. This option is required if you did not select *Use TNS name*.
TNS name	Refer to the requirements of your database	Enter an existing Oracle Transparent Network Substrate (TNS) name through which the software accesses sources and targets defined in this datastore. This option is required when you select *Use TNS name*.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which the software accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.
Enable CDC	-	Select to enable changed data capture for this datastore.

Table 76: General

Oracle option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default

Oracle option	Possible values	Description
		commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click *Browse*	Enter the location where command and data files are written for bulk loading. For Solaris systems, the path name must be less than 80 characters. You can enter a variable for this option.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. You can enter a variable for this option.

Table 77: Locale

Oracle option	Possible values	Description
Language	-	See the section "Locales and Multi-Byte Functionality."
Code page	-	See the section "Locales and Multi-Byte Functionality."
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 78: Session

Oracle option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon	Additional session parameters specified as valid SQL statement(s)

Table 79: Oracle Miscellaneous

Oracle option	Possible values	Description
Default precision for number	1 <= precision <= 96	Enter the total number of digits in the value.
Default scale for number	0 <= scale <= precision	Enter the number of digits to the right of the decimal point.

Table 80: Aliases (Click here to create)

Oracle option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

Table 81: Linked Datastores (Click here to create)

Oracle option	Possible values	Description
Datastore Name	Alphanumeric characters and underscores or blank	The name of a datastore to which you linked the current datastore configuration in preparation to import a database link

3.2.2.7.2.12 Persistent Cache

Table 82: Locale

Persistent Cache option	Possible values	Description
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 83: Session

Persistent Cache option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon	Additional session parameters specified as valid SQL statement(s)

3.2.2.7.2.13 SAP HANA

Option	Possible values	Description
Database version	SAP HANA database `<version number>`	Select the version of your SAP HANA database client. This is the version of the SAP HANA database that this datastore accesses.
Use data source name (DSN)	Check box selected or not selected	Select to use DSN to connect to the database. By default, this option is not selected and a server name (also known as DSN-less) connection will be used. For a DSN-less connection, you must fill in *Database server name* and *Port*. If you select this checkbox, you must fill in *Data source name*.
Database server name	Computer name	Enter the name of the computer where the SAP HANA server is located. This option is required if you did not select *Use data source name (DSN)*.
Port	Five digit integer Default: 30015	Enter the port number to connect to this SAP HANA Server. This option is required if you did not select *Use data source name (DSN)*.
Data source name	Refer to the requirements of your database	Select or type the Data Source Name defined in the ODBC Administrator for connecting to your database. This option is required when you select *Use data source name (DSN)*.

Option	Possible values	Description
User name	Alphanumeric characters and underscores	Enter the user name of the account through which the software accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.
Enable automatic data transfer		The *Enable automatic data transfer* check box is selected by default when you create a new datastore and you choose Database for *Datastore type*. Keep *Enable automatic data transfer* selected to enable transfer tables in this datastore that the Data_Transfer transform can use to push down subsequent database operations.
Additional connection information	Alphanumeric characters and underscores, or blank	Enter information for any additional parameters that the data source supports (parameters that the data source's ODBC driver and database support). Use the format: `<parameter1=value1; parameter2=value2>`
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Overflow file directory	Directory path or click Browse.	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.
Aliases (Click here to create)	-	Enter the alias name and the owner name to which the alias name maps.

3.2.2.7.2.14 SAP Sybase ASE

Table 84: Main window

SAP Sybase ASE option	Possible values	Description
Database version	SAP Sybase ASE **`<version number>`**	Select the version of your SAP Sybase ASE client. This is the version of SAP Sybase that this datastore accesses.
Database server name	Computer name	Enter the name of the computer where the SAP Sybase ASE instance is located.

SAP Sybase ASE option	Possible values	Description
		Note For UNIX Job Servers, when logging in to a SAP Sybase repository in the Designer, the case you type for the database server name must match the associated case in the SYBASE_Home\interfaces file. If the case does not match, you might receive an error because the Job Server cannot communicate with the repository.
Database name	Refer to the requirements of your database	Enter the name of the database to which the datastore connects.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which the software accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.

Table 85: General

SAP Sybase ASE option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.

Table 86: Locale

SAP Sybase ASE option	Possible values	Description
Language	-	See the section "Locales and Multi-Byte Functionality."
Code page	-	See the section "Locales and Multi-Byte Functionality."
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 87: Session

SAP Sybase ASE option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon	A valid SQL statement or multiple SQL statements delimitated by semicolon.

Table 88: Aliases (Click here to create)

SAP Sybase ASE option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

3.2.2.7.2.15 SAP Sybase IQ

Displayed options vary depending on the version of SAP Sybase IQ you select in the datastore editor.

Table 89: Main window

SAP Sybase IQ option	Possible values	Description
Database version	Currently supported versions	Select the version of SAP Sybase IQ that this datastore accesses. Displayed options in the rest of the datastore editor vary depending on the version selected.
Use data source name (DSN)	Check box selected or not selected	Select to use DSN to connect to the database. By default, this option is not selected and a server name (also known as DSN-less) connection will be used. For a DSN-less connection, you must fill in *Hostname*, *Database name*, and *Port*, and *Server name*. If you select this checkbox, you must fill in *Data source name*
Hostname	Computer name or IP address	Type the computer name or IP address. This option is required if you did not select *Use data source name (DSN)*.
Database name	Refer to the requirements of your database	Type the name of the database defined in SAP Sybase IQ. This option is required if you did not select *Use data source name (DSN)*.
Port	Four digit integer Default: 2638	Type the number of the database port. This option is required if you did not select *Use data source name (DSN)*.
Server name	Refer to the requirements of your database	Type the SAP Sybase IQ database server name. This option is required if you did not select *Use data source name (DSN)*.
Database server name	Refer to the requirements of your database	Type the SAP Sybase IQ database server name.

SAP Sybase IQ option	Possible values	Description
		This option is required if you did not select *Use data source name (DSN)*.
Data source name	Refer to the requirements of your database	Select or type the Data Source Name defined in the ODBC Administrator for connecting to your database. This option is required when you select *Use data source name (DSN)*.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which the software accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.

Table 90: General

SAP Sybase IQ option	Possible values	Description
Rows per commit	Positive integer	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click *Browse*	Enter the location where command and data files are written for bulk loading. For Solaris systems, the path name must be fewer than 80 characters. You can enter a variable for this option. If you do not enter a name here, Data Services by default writes the files to the directory **`<DS_COMMON_DIR>`**`\log\bulkloader`.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. You can enter a variable for this option.

Table 91: Locale

SAP Sybase IQ option	Possible values	Description
Language		See the section "Locales and Multi-Byte Functionality."
Code page		See the section "Locales and Multi-Byte Functionality."
Server code page		See the section "Locales and Multi-Byte Functionality."

Table 92: Bulk loader

SAP Sybase IQ option	Possible values	Description
JS and DB on same machine	Yes, No	For some versions of SAP Sybase IQ, you must configure Data Services to transfer via FTP the data file generated on the Job Server (JS) to the database server (DB). Refer to the following table for how to use this option together with the *Use named pipe* option and FTP depending on the version of SAP Sybase IQ you are using.
Server working directory	Absolute file path	Type the path to the working directory for the load utility on the computer that runs the SAP Sybase IQ server. If an absolute file path is specified for the *FTP host working directory* box, then the Server working directory is optional. If the box is left blank, then the software uses the file path to the *FTP host working directory*. If a virtual file path is specified for the *FTP host working directory*, then you must enter an absolute file path in the *Server working directory* box.
Use named pipe	Yes, No	Select Yes to eliminate the need to write a data file to disk, which can improve performance. If a data file is required for SAP Sybase IQ database recovery, select No. Defaults to No. Refer to the following table for how to use this option together with the *JS and DB on same machine* option and FTP depending on the version of SAP Sybase IQ you are using.

Table 93: Supported bulk loader options and methods

JS and DB on same machine	Use named pipe	For SAP Sybase IQ versions earlier than 15.x	For SAP Sybase IQ versions 15.x and later
Yes	Yes	Named pipe	Named pipe
Yes	No	File	File
No	Yes	FTP	Named pipe
No	No	FTP	File

Table 94: FTP

SAP Sybase IQ option	Possible values	Description
FTP host name	Computer name, fully qualified domain name, or IP address	For some versions of SAP Sybase IQ, Data Services generates a data file and transfers it via FTP to the database for loading.

SAP Sybase IQ option	Possible values	Description
		Type the name of the SAP Sybase IQ server computer (host). If the FTP host name is left blank and Data Services needs this FTP information for bulk loading, it generates a validation error.
FTP login user name	Alphanumeric characters and underscores, or blank	Must be defined to use FTP.
FTP login password	Alphanumeric characters, underscores, and punctuation, or blank	Must be defined to use FTP.
FTP host working directory	Absolute file path Virtual file path (Windows servers only)	The location on the database server where Data Services transfers the data file between the Job Server and the SAP Sybase IQ server. For Windows servers only, you can configure a path to a virtual directory.

Table 95: Session

SAP Sybase IQ option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolon	Additional session parameters specified as valid SQL statement(s).

Table 96: SAP Sybase Miscellaneous

SAP Sybase IQ option	Possible values	Description
Use linked remote servers	Yes, No	This option lets you use the INSERT...LOCATION SQL statement for a data flow that uses SAP Sybase IQ as the loader and SAP Sybase ASE or SAP Sybase IQ as the reader. The Data Services engine pushes down the SQL statement for the SAP Sybase IQ server location. Select Yes to use remote servers that have already been linked. To define a remote server, use the CREATE SERVER SQL statement in SAP Sybase IQ. To set up the remote login for users, use the CREATE EXTERNLOGIN SQL statement. For detailed information about the SQL statements, see the relevant SAP Sybase IQ product documentation.

Table 97: Aliases (Click here to create)

SAP Sybase IQ option	Possible values	Description
Aliases		Enter the alias name and the owner name to which the alias name maps.

3.2.2.7.2.16 Teradata

Table 98: Main window

Teradata option	Possible values	Description
Database version	Teradata `<version number>`	Select the version of your Teradata client. This is the version of Teradata that this datastore accesses.
Use data source name (DSN)	Check box selected or not selected	Select to use DSN to connect to the database. By default, this option is not selected and a server name (also known as DSN-less) connection will be used. For a DSN-less connection, you must fill in *Database server name*, *Database name*, and *Port*. If you select this checkbox, you must fill in *Data source name*
Database server name	Refer to the requirements of your database	Type the Teradata database server name. This option is required if you did not select *Use data source name (DSN)*.
Database name	Refer to the requirements of your database	Type the name of the database defined in Teradata. This option is required if you did not select *Use data source name (DSN)*.
Port	Four digit integer Default: 8888	Type the port number to connect to this database. This option is required if you did not select *Use data source name (DSN)*.
Data source name	Refer to the requirements of your database	Type the Data Source Name defined in the ODBC Administrator for connecting to your database. This option is required when you select *Use data source name (DSN)*.
User name	Alphanumeric characters and underscores	Enter the user name of the account through which the software accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.

Table 99: General

Teradata option	Possible values	Description
Bulk loader directory	Directory path or click *Browse*	Enter the location where command and data files are written for bulk loading. For Solaris systems, the path name must be less than 80 characters. You can enter a variable for this option.
Bulk reader directory	Directory path or click *Browse*	Specify the directory on the Job Server where Teradata Parallel Transporter-specific files (con-

Teradata option	Possible values	Description
		trol and error files) are stored. If this option is left empty, the default location is `$LINK_DIR\log\BulkReader`.
Overflow file directory	Directory path or click *Browse*	Enter the location of overflow files written by target tables in this datastore. You can enter a variable for this option.

Table 100: Locale

Teradata option	Possible values	Description
Language		See the section "Locales and Multi-Byte Functionality."
Code page		See the section "Locales and Multi-Byte Functionality."
Server code page		See the section "Locales and Multi-Byte Functionality."
Teradata		
Log directory	Directory path or click *Browse*	The directory in which to write log files.
TdpId	Alphanumeric characters, underscores, and punctuation	Teradata Director Program Identifier which identifies the name of the Teradata database to load. If you use bulk loading, this identifier is mandatory.

Table 101: Session

Teradata option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolons.	Additional session parameters specified as valid SQL statement(s). For example, to use the Table_Comparison transform with Teradata 13 and later tables as the comparison table and target table, you must do the following: ● On the Teradata server, set the *General* parameter *DBSControl* to TRUE to allow uncommitted data to be read. ● In the Data Services Teradata datastore, add the following statement in the "Additional session parameters" field: `SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL READ UNCOMMIT-TED;`

Table 102: Aliases (Click here to create)

Teradata option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

3.2.2.7.3 Application datastores

The information you enter for a datastore depends on the type of datastore to which you are connecting. Application datastore types include:

Datastore type	More information
JDE OneWorld	Datastore configuration options for this datastore type vary depending on which database type you select. The JDE OneWorld datastore type works with the following database types: ● DB2 ● Microsoft SQL Server ● ODBC ● Oracle For details on configuring the datastore options for JD Edwards applications, refer to the *Supplement for J.D. Edwards*.
JDE World	For JDE World, Data Services supports the database type ODBC. For details on configuring the datastore options for JD Edwards , applications refer to the *Supplement for J.D. Edwards*.
Oracle Applications	For details on configuring the datastore options for Oracle applications, refer to the *Supplement for Oracle Applications*.
PeopleSoft	Datastore configuration options for this datastore type vary depending on which database type you select. The PeopleSoft datastore type works with the following database types: ● Microsoft SQL Server ● Oracle For more information about PeopleSoft applications, refer to the *Supplement for PeopleSoft*.
SAP Applications	The options for all SAP datastore types (SAP Applications, SAP BW Source, SAP BW Target, and SAP Master Data Services) are documented in the *Supplement for SAP*.
SAP BW Source	
SAP BW Target	
SAP Master Data Services	
Siebel	The Siebel datastore type works with the following database types:

Datastore type	More information
	• DB2 • Microsoft SQL Server • Oracle For more information about Siebel applications, refer to the *Supplement for Siebel*.
Web Service	The options for the Web Service datastore type are documented in the *Integrator Guide*.

After you create a datastore, you can import metadata about the objects, such as tables and functions, into that datastore.

Related Information

Designer Guide: Datastores, What are datastores? [page 209]
Supplement for JDEdwards: Datastores [page 2354]
Supplement for Oracle Applications: Datastores [page 2363]
Supplement for PeopleSoft: PeopleSoft Datastores [page 2368]
Supplement for PeopleSoft: Reference Information, Datastore [page 2380]
Supplement for SAP: Reference Information, Datastore [page 2531]
Supplement for SAP: Connecting to SAP Applications, SAP Applications datastores [page 2438]
Supplement for SAP: Connecting to SAP NetWeaver Business Warehouse, SAP NetWeaver BW datastores [page 2508]
Supplement for Siebel: Datastores [page 2584]
Integrator Guide: Consuming external web services in SAP Data Services [page 2257]

3.2.2.8 Document

Class

Reusable

Access

In the object library, click the *Datastores* tab.

Description

Available in some adapter datastores, documents describe a data schema. Documents can support complicated nested schemas. You can use documents as sources or targets.

See your adapter's documentation for more specific information about the options available for documents.

3.2.2.9 DTD

Class

Reusable

Access

In the object library, click the *Formats* tab, then open the DTD category.

Description

A DTD (document type definition) describes the data schema of an XML message or file.

> **Note**
>
> XML Schemas can be used for the same purpose.

Data flows can read and write data to messages or files based on a specified DTD format.You can use the same DTD to describe multiple XML sources or targets.

To use DTDs, import metadata into SAP Data Services. You can import a DTD directly, or you can import an XML document that contains or references a DTD. During import, the software converts the structure defined in the DTD into the nested-relational data model (NRDM).

Related Information

XML schema [page 1008]
Rules for importing DTDs [page 911]

3.2.2.9.1 Editor

Open the DTD editor by double-clicking a DTD name in the object library.

Related Information

Designer Guide: Using Document Type Definitions (DTDs) [page 353]

3.2.2.9.2 Properties

DTDs have the following properties.

Property	Description
Name	The name of the format. This name appears in the object library under the Formats tab and is used for sources and targets (XML files or messages) that reference this format in data flows.
Description	Text that you enter to describe and document the DTD.
Imported from	The full path to the format. For example, `C:\data\test.dtd.` A variable can also be used.
DTD file	(Read-only) If the check box is selected, the DTD format was originally imported from a DTD file. Otherwise, it was imported from an XML file with an associated DTD.
Root element name	The name of the primary node of the XML that the DTD is defining. SAP Data Services only imports elements of the format that belong to this node or any sub nodes.

3.2.2.9.3 Attributes for DTDs

The following DTD attributes are supported.

Supported column attribute	Description
Enumeration	Contains a list of all possible values separated by vertical bars. For example: "Red | White | Blue Green | Magenta". A string display is cut off at 256 characters.
Fixed Value	The only value the column can have.
Native Type	String. The original data type of the of the element or attribute in the DTD.
Required	Indicates whether this column always has to be mapped (YES/NO). If a column is optional (required =no), then validation will allow mapping expressions to be missing for these columns and at runtime the engine will substitute NULLs for the missing values.
XML Type	Allows you to track whether the column was an element or attributes in the original DTD.

Supported nested table attribute	Description
Any One Column	If choice (for example, "white | black | almond"), then SAP Data Services sets the value of Any One Column to YES. If sequence (for example, "first, last, street, city, state") then the software sets the value to NO. If both are present in the DTD, the value is set to NO.
Minimum Occurrence	If ()* then minimum occurrence is set to zero. If ()+, then minimum occurrence is set to 1. Indicates minimum number of rows that can be in the table.

3.2.2.9.4 Supported DTD components

SAP Data Services reads the following DTD components. To process the data read in an XML file or message, the software translates the DTD into its internal nested-relational database model.

Each component in the DTD is defined by its `<content model>`. The software supports the declarations in XML content models as follows:

DTD declaration		Supported
DOCTYPE	SYSTEM	Supported.
	PUBLIC	No support.
Declarations	ELEMENT	Supported. The XML Type attribute of the corresponding column is set to Element.
	ATTRIBUTE	Supported. The XML Type attribute of the corresponding column is set to Attribute.
	ENTITY	Supported. All entity references that can be expanded are expanded. Any that cannot be expanded cause an error at the time that you import the DTD.
	NOTATION	No support. Elements defined with NOTATION cause an error at the time that you import the DTD.
Content model	ANY	No support. Elements defined with ANY cause an error at the time that you import the DTD.
	EMPTY	Supported.
	#PCDATA	Supported. Converts to varchar(1024).
	MIXED	Supported.
Attribute declarations	CDATA	Supported. Converts to varchar(1024).
	ID	Supported. Converts to varchar(1024). When producing XML output, the software cannot ensure that ID values are unique throughout the schema.
	IDREF	Supported. Converts to varchar(1024).
	IDREFS	Supported. Converts to varchar(1024).
	NMTOKEN	Supported. Converts to varchar(1024).
	NMTOKENS	Supported. The software treats multiple tokens as a single token with more than one space-separated values.
	Enumerated value	Supported. The software saves the enumerated values in the Enumeration attribute of the column. When producing XML output, the software checks to ensure that the value generated by the real-time job for the corresponding column is from the list; if no value is generated, the software uses the provided default value. If you validate XML messages against the DTD in a real-time job and the message includes a value that is not allowed based on the DTD, the XML source produces an error.
Attribute declaration defaults	#REQUIRED	Supported. The software saves this as the Required attribute with a value of `YES` and as data type `varchar(1024)`. When pro-

DTD declaration		Supported
		ducing XML output, the software always provides a value. If there is no value supplied, the output value is NULL (`''`).
	#IMPLIED	Supported. The software saves this as the Required attribute with a value of `NO` and as the data type `varchar(1024)`. When producing XML output, the software provides whatever value is generated in the data flow for the corresponding column, including a NULL value (`''`).
	#FIXED (default value)	Supported. The software saves this as the Fixed Value attribute and the data type `varchar(1024)`. When producing XML output, the software checks to ensure that the value generated by the real-time job for the corresponding column is from the list; if no value is generated, the software uses the provided default value.
	Default values	Supported. Converts to data type `varchar(1024)`. When producing XML output, the software uses the default value if the value generated in the real-time job for the corresponding column is NULL.

To produce a data model that can include all possible configurations of an element, the software can simplify some of the content model operations:

Operator	Description	Supported
No operator	One and only one	One and only one.
Comma (,)	Sequence	Supported. The software uses the ordering given in the DTD as the column ordering in the internal data set. Also the Any One Column attribute is set to a value of `NO`.
Vertical bar (|)	Choice (either/or)	Supported. The software uses the ordering given in the DTD as the column ordering in the internal data set. Also the Any One Column attribute is set to a value of `YES`. The internal data set must include both options.
Plus (+)	One or more	Supported. Saved as nested table attribute Minimum Occurrence with a value of "1". The internal data set must include options for one or more elements.
Asterisk (*)	Zero or more	Supported.

Operator	Description	Supported
		Saved as nested table attribute Minimum Occurrence with a value of "0". The software translates an item or grouping including zero or more items into a nested table.
Question mark (?)	Optional	Supported. The internal data set includes the Required attribute set to a value of NO for the corresponding column or nested table.
Parentheses ()	Group	Dropped. The internal data set does not maintain groupings unless the group is operated on by the * operator. If the group can allow more than one item, the software makes a new nested table into which it places the elements in the group.

After these simplifications, the software needs only work with two DTD operators: sequence (strict ordering) and the combined operators of the group operator with the zero or more item operator. For the purpose of representing the data internally in the software, all DTDs can now be written using only , or ()*.

3.2.2.9.5 Rules for importing DTDs

SAP Data Services applies the following rules to convert a DTD to an internal schema:

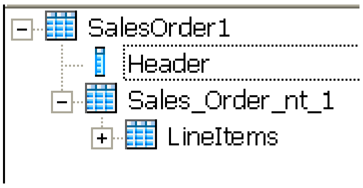
- Any element that contains an PCDATA only and no attributes becomes a column.
- Any element with attributes or other elements (or in mixed format) becomes a table.
- An attribute becomes a column in the table corresponding to the element it supports.
- Any occurrence of choice (,)or sequence (|) operators uses the ordering given in the DTD as the column ordering in the internal data set.
- Any occurrence of a multiple entities, such as ()* or ()$^{+}$, becomes a table with an internally generated name (an implicit table).
- The internally generated name is the name of the parent followed by an underscore, then the string "nt" followed by a sequence number. The sequence number starts at 1 and increments by 1.

After applying these rules, the software uses two additional rules, except where doing so would allow more than one row for a root element:

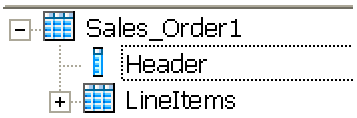
- If an implicit table contains one and only one nested table, then the implicit table can be eliminated and the nested table can be attached directly to the parent of the implicit table.
  For example, the SalesOrder element might be defined as follows in a DTD:

  ```
  <!ELEMENT SalesOrder (Header, LineItems*)>
  ```

  When converted into the software, the LineItems element with the zero or more operator would become an implicit table under the SalesOrder table. The LineItems element itself would be a nested table under the implicit table.



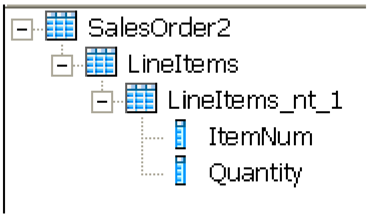
Because the implicit table contains one and only one nested table, the format would remove the implicit table.



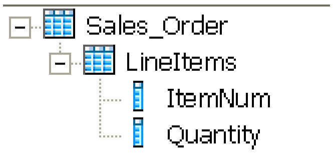
- If a nested table contains one and only one implicit table, then the implicit table can be eliminated and its columns placed directly under the nested table.
  For example, the nested table LineItems might be defined as follows in a DTD:

  ```
  <!ELEMENT LineItems (ItemNum, Quantity)*>
  ```

  When converted into the software , the grouping with the zero or more operator would become an implicit table under the LineItems table. The ItemNum and Quantity elements would become columns under the implicit table.

  Because the LineItems nested table contained one and only one implicit table, the format would remove the implicit table.

3.2.2.9.6 Design considerations

The following areas provide opportunities for you to improve performance and tune the nested-relational data model results for a given DTD:

- Recursion
  If the DTD contains an element that uses an ancestor element in its definition, SAP Data Services expands the definition of the ancestor for a fixed number of levels. For example, given the following definition of element "A":
  A: B, C
  B: E, F
  F: A, H

The software produces a table for the element "F" that includes an expansion of "A." In this second expansion of "A," "F" appears again, and so on until the fixed number of levels. In the final expansion of "A," the element "F" appears with only the element "H" in its definition.

- Repeated column names
  The software does not allow more than one column with the same name at the same level in a table. If the internal schema that the software produces from a DTD contains duplicate names, the software adds a suffix to each instance of the duplicated name to ensure unique column names. The suffix is an underscore followed by a sequence number, starting at 1 and incrementing by 1.
  A DTD can produce duplicate names when the DTD contains a repeated element at one level or an element with a scalar value with an attribute of the same name.
- Ambiguous DTDs
  You can create a DTD such that the software does not have enough information to make a unique decision when generating the internal data set. The software reacts to an ambiguous DTD by throwing an error for the XML message source at run time. An example of an ambiguous definition is as follows:

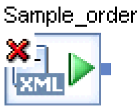
DTD	`A: ((B, (C)*) | (B, (D)*))+`
Schema in Data Services	`A: (B, (C)*, B, (D)*)*`

XML input	`A: <B>text</B>`
	`<D>1</D> <D>2</D>`

The software will use the B element data to populate the first B column, then use the D element data to populate the D element. If this data is then translated back into XML, it would be invalid relative to the DTD.

Metadata

If you delete a DTD from the object library, XML file and message sources or targets that are based on this format are invalid. The software marks the source or target objects with an icon that indicates the calls are no longer valid.



To restore the invalid objects, you must delete the source or target and replace it with a source or target based on an existing DTD.

3.2.2.9.7 Error checking

You can control whether SAP Data Services checks each incoming XML file or message for validity. If you choose to check each XML file or message, the software uses the DTD imported and stored in the repository rather than a DTD specified by a given XML file or message. If a file or message is invalid relative to the DTD, the real-time job produces an error and shuts down.

During development, you might validate all files and messages to test for error conditions. During production, you might choose to accept rare invalid files or messages and risk ambiguous or incorrect data.

All files or messages that the software produces for an XML file or message target are validated against the imported DTD.

You can enable or disable validation for an XML file or message source or target in that object's editor.

3.2.2.10 Excel workbook format

Class

Reusable

Access

In the object library, click the *Formats* tab.

Description

An Excel workbook format describes the structure defined in an Excel workbook (denoted with a .xls extension). You store format templates for Excel data ranges in the object library. You use the template to define the format of a particular source in a data flow. SAP Data Services accesses Excel workbooks as sources only (not as targets).

You can define the format schema by:

- Import the Excel metadata from a sample Excel workbook and create the schema automatically. You can opt to import the schema from:
    - A named range defined in an Excel workbook.
    - A custom range in a worksheet (for example A1:C10).
    - All fields. Note that the field range does not necessarily begin on row 1, column 1; it starts at the upper left-most cell in the worksheet that contains data.

  After you select an access method, import the schema by clicking Import schema.

  > i Note
  >
  > Importing overwrites the existing schema.

After you import, you can edit column names, data types, and content types in the schema pane at the top of the window. You can optionally add descriptions for each column.

- You can manually define (enter) column names, data types, content types and descriptions in the schema pane at the top of the window.

You will see Blank if the content type cannot be automatically determined for a column.

3.2.2.10.1 Notes

- If SAP Data Services cannot determine the data type for a column, for example if your column selection doesn't contain any data, it imports the column as varchar(255).
- If the worksheet is empty and you select the *All fields* option, the software creates a single field, F1 varchar(255).
- For workbook-specific (global) named ranges, the software would name a range called *range* as *range*. However for worksheet-specific (local) named ranges, the software would name a range called *range* that belongs to the worksheet *Sheet1* as *range!Sheet1*.
  In UNIX, you must also include the worksheet name when defining a workbook-specific (global) named range.
- Because the software reads stored formula values, incorrect values could result if Excel does not exit properly. Close and reopen the file in Excel to refresh the values.
- If an invalid Excel formula displays an error such as #DIV/0, #VALUE or #REF!, the software processes the cell as NULL.
- You cannot import or process a password-protected workbook.
- The software might not be able to import or process blank worksheet names or those that contain special characters such as /?:!*[]`.

3.2.2.10.2 Import or Edit Excel workbook format options

The top panel of the *Import* or *Edit Excel Workbook* window displays:

- *Format name:* The name of the Excel workbook format template in the object library. In the *Import Excel Workbook* window, you can define the name of the format.
- *Schema pane*: This pane lets you manually define (or edit) the schema of the Excel workbook template.

The Import (and Edit) Excel workbook format dialog boxes include options on the *Format* and *Data Access* tabs.

Format tab

The *Format* tab defines the parameters of the Excel workbook format.

Format option	Description
Directory	Specify the directory that contains the Excel workbook. You can specify variables or wild cards, but the *Import schema* button will then be disabled.
File name	Specify the file name for the Excel workbook file. This file contains the schema definition. You can specify variables or wild cards, but the *Import schema* button will then be disabled.
Access method	Specifies whether to import a named range (as defined in Excel) or a range from a worksheet: *Named range*—The drop-down box displays named ranges defined in the Excel workbook (provided that the directory/file name combination refers to a valid file). You can also type the name manually. Note The drop-down does not display non-contiguous named ranges, even if they exist in the Excel workbook. *Worksheet*—You can select a specific worksheet by either name or ordinal number (select the *Number* check box to designate the worksheet name as a number). The drop-down box displays all worksheets in the Excel workbook (provided that the directory/filename combination refers to a valid file). You can also type the name manually. Note If the worksheet name starts with a dollar sign ($), Data Services treats it as a variable; to use a worksheet name that starts with a dollar sign, prefix (escape) it with a backslash (\). The Worksheet method includes the following *Range* options: ● *All fields*—Includes everything from the uppermost left-hand populated cell to the lowest, right-hand cell. ● *Custom range*—Uses Excel notation, for example A1:B3. Click the button to the right of the field to launch a new instance of Excel (if installed) to select the cell range directly in the worksheet. Note Subsequently making the software the active window closes this Excel instance. *Extend range*—If you are using a custom range, select this check box to extend the custom range selection to the end of the worksheet.

Format option	Description
Code page	Specifies the character encoding (code page) of the character data in the Excel workbook. For more information, see the Locales and Multi-Byte Functionality section in the *Reference Guide*.
Use first row values as column names	Select to use the first-row values as column names. If this check box is cleared, then the software names the fields for you (F1 represents the first field, F2 represents the second field, and so on).

Data Access tab

The Data Access tab specifies how the software accesses the data file. If both the *FTP* and *Custom* check boxes are cleared, the software assumes the data file is on the same computer as the Job Server.

Data access option	Description
FTP	Select to use FTP to access the data file.
Host	Type the computer (host) name, fully qualified domain name, or IP address of the computer where the data file resides.
User	Type the FTP user name.
Password	Type the FTP user password.
Directory	Type or browse to the directory that contains the Excel workbook data file to import. If you include a directory path here, then enter only the file name in the *Name* field.
File name	Type or browse to the Excel workbook data file *Name*. You can use variables or wild cards (* or ?). If you leave *Directory* blank, then type a full path and file name here.
Custom	Select to use a custom executable to access the data file.
Executable	Type the name of the program to read data file.
User	Type the user name.
Password	Type the password.
Arguments	Include any custom program arguments.

3.2.2.10.3 Excel workbook source options

The source editor includes the following Excel workbook options on the following tabs. Note that many fields in the source editor allow you to select from a list of variables.

- Source tab
- Format tab
- Data Access tab

Source tab

Source option	Description
Make port	Makes the source table an embedded data flow port.

Table 103: Performance

Source option	Description
Join rank	Indicates the rank of the source relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. For new jobs, specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For more information, see the Other Tuning Techniques section in the *Performance Optimization Guide*.
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. Options are: • *Yes*: The source is always cached unless it is the outer-most source in a join. • *No*: The source is never cached. The default is *Yes*. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. For new jobs, specify the cache only in the Query transform editor.
Skip all empty rows	Select to skip any rows that are empty in the workbook. Clear to also import empty rows, which will display as NULLs.

Table 104: Error handling

Source option	Description
Log errors to file	Specifies whether to produce an error report. Defaults to unchecked. Columns with run-time conversion errors will contain NULLs.
Maximum errors to log	If *Log data conversion warnings* is enabled, you can limit how many warnings the software logs. Defaults to *{no limit}*.
Error file directory	Type or browse to the directory in which to store the error file.
Error file name	Type or browse to the error file name.

Table 105: Include file name column

Source option	Description
Include file name column	Determines whether to add a column that contains the source Excel workbook file name in the source output. Defaults to *No*. Change the value to *Yes* when you want to identify the source Excel workbook file in situations such as the following: ● You specified a wildcard character to read multiple source Excel workbook files at one time ● You load from different source workbook files on different runs
Modify	If the file name is included, this button enables you to modify *File name column* and *Column size*.
File name column	If the file name is included, the name of the column that holds the source Excel workbook file name. Defaults to *DI_FILENAME*.
Column size	If the file name is included, the size (in characters) of the column that holds the source Excel workbook file name. Defaults to *100*. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.
Include path	If the file name is included, determines whether to include the full path name of the source Excel workbook file. Defaults to *No*.

Table 106: Other options

Source option	Description
Skip first	Optionally enter the number of rows to skip (not read) starting at the top of the worksheet. Defaults to *{none}*.
Read total	Optionally enter the number of total rows to read starting at the top of the worksheet or after the *Skip first* value. Defaults to *{no limit}*.

Related Information

Performance Optimization Guide: Other tuning techniques, Join ordering [page 2198]
Designer Guide: Embedded Data Flows [page 391]

3.2.2.11 File format

Class

Reusable

Access

In the object library, click the *Formats* tab.

Description

A file format describes the structure of an ASCII file. You store templates for file formats in the object library. You use the templates to define the file format of a particular source or target file in a data flow.

A file format consists of multiple properties. You set the properties in the *File Format Editor*. Available properties vary by the mode of the *File Format Editor*. The modes are as follows:

Mode	Description
New	Use to create a new file format template. To open in new mode, go to the *Formats* tab in the object library, right-click *Flat Files*, and select *New*. The *File Format Editor* appears.
Edit	Use to edit an existing file format template. To open in edit mode, go the *Formats* tab in the object library, select an existing flat file format, double-click, or right-click and select *Edit*. The *File Format Editor* appears.
Source	Use to edit the file format of a particular source file. To open in source mode, click the name of the source file in the workspace. The *File Format Editor* appears below the file's schema.

Mode	Description
Target	Use to edit the file format of a particular target file. To open in target mode, click the name of the target file in the workspace. The *File Format Editor* appears below the file's schema.

The work area on the left, in the *File Format Editor* lists file format properties that are not field specific. The following table lists all of the options. These options are filtered by the mode you are using.

Option	Possible values	Description	Mode
General			
Type	Delimited, Fixed width, SAP transport, Unstructured text, Unstructured binary	The format of the data in the text file. Available properties change based on the selected file format type. For information about the SAP transport file format, see the *Supplement for SAP*.	New, Edit
Name	Any alphanumeric character and underscores	A descriptive name for the file format. This name appears in the object library.	New
Join rank	Integer greater than or equal to 0	Indicates the rank of the source relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. For new jobs, specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For more information, see the “Other Tuning Techniques” section in the *Performance Optimization Guide*.	Source
Cache	Yes, No	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. Options are: ● *Yes*: The source is always cached unless it is the outer-most source in a join. ● *No*: The source is never cached. The default is *Yes*.	Source

Option	Possible values	Description	Mode
		Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. For new jobs, specify the cache only in the Query transform editor.	
Adaptable schema	Yes, No	Indicates whether the schema of a delimited file format is adaptable or fixed. ● *Yes*: Indicates that the schema is adaptable. The actual file can contain fewer or more columns than indicated by the file format. If a row contains fewer columns than expected, the software loads null values into the columns missing data. If a row contains more columns than expected, the software ignores the additional data. ● *No*: Indicates that the schema is fixed. The software requires the number of columns in each row to match the number of columns specified in the file format. The default is *No*. If you select *Yes*, you must ensure that the selected column and row delimiters do not appear inside the actual data.	New, Edit, Source
Data Alignment	Character, Byte	Indicates how the software will process fixed-width file formats. The default setting is *Character*. *Character* indicates that fields in your data are measured as character. All processing will be in character semantics. *Byte* indicates that fields in your data are measured as bytes. All processing will be in byte semantics. For example, in character semantic, if you define a column as varchar(30), it means that the column has 30 characters regardless of the number of bytes for each character. In byte semantic, varchar(30) means 30 bytes. Byte semantic recognizes bytes only, it does not recognize characters. Note The reader reads as many bytes in a column based on the length of the column. For multibyte code pages, a character can be more than one byte. There could be times when trailing bytes of the last character of a column exceeds the length of the column. When this happens the reader continues reading until it gets to the total length of the character. Also in this situation, the loader writes as many bytes as what is specified for the length of the column. The loader then truncates trailing bytes if they exceed the length of the column and the code page is multibyte. Therefore there is a potential for partial characters in the file. When this happens, errors are only logged in the reader. You can specify the error logging options in the *File Format Editor*.	New, Edit

Option	Possible values	Description	Mode
Make port	Yes, No	Indicates whether the file is an embedded data flow port. Choose *Yes* to make a source or target file an embedded data flow port. The default is *No*. For more information, see "Creating embedded data flows" in the *Designer Guide*.	Source, Target
Rows to read	Integer or blank	Indicates the maximum number of rows that the software reads. The default is blank. If the value is zero or negative, the software reads all rows.	Source
Custom transfer program	Yes, No	Enables the software to use a third-party file transfer program. Displays additional Custom transfer program options in the *File Format Editor* below the Input/Output properties.	All
Skip error handling	Yes, No	Selecting *Yes* disables the Error Handling section in the Format editor. The default is *No.*	New, Edit, Source
Parallel process threads	Integer greater than 0, {none}, {default}	Specifies the number of threads for parallel processing, which can improve performance by maximizing CPU usage on the Job Server computer. For example, if you have four CPUs, enter 4 for this option. For more information, see "File multi-threading" in the *Performance Optimization Guide*. For jobs that process USPS certification tests, the value should be set to *{none}*.	All
Data File(s)			
Location	Local, Job Server	During design, indicates whether files are located on the local machine that runs the Designer or on the machine that runs the Job Server. If you select Job Server, you must enter the absolute path to files. Remember that UNIX systems are case-sensitive. During execution, all files must be located on the Job Server machine that executes the job. If you use different files to design your job, change the file specified (through the Root directory and File properties) before execution.	All
Root directory	Path name for the file or blank	The directory where the file is located. For added flexibility, you can enter a variable for this option. If you enter a directory name, then enter only the file name for the *File* property. If you leave the root directory blank, then enter a file name that includes the full path name in the *File* property.	New, Edit
File name(s)	File name(s), file name(s) including full	In new and edit modes, specifies an existing file on which you base the file format description. Data from this file appears in the Col-	All

Option	Possible values	Description	Mode
	path name, or blank	umn Attributes area. In these modes, you can leave this property blank. In source and target modes, specify the location of the actual file for this source or target. In these modes, you cannot leave this property blank. For added flexibility, you can enter: ● A variable that is set to a particular file with full path name. Use variables to specify file names that you cannot otherwise enter, such as file names that contain multi-byte characters. ● A list of files, separated by commas, or a file name containing a wild card. In this case, the software reads all these files as a single source. See “Reading multiple files at one time” in the *Designer Guide*.	
Read subfolders	Yes, No	For unstructured file formats, specifies whether to read the files in any and all nested subfolders.	New, Edit, Source
Skip empty files	Yes, No	For unstructured file formats, specifies whether to ignore empty files. ● *Yes*: Skips empty files. ● *No*: Creates a row with NULL data.	New, Edit, Source
Number of files to read	Integer or {none}	For unstructured file formats, indicates the maximum number of files to read. A zero or negative value reads all files. The default is {none}.	Source
Delete file	Yes, No	Specifies whether the software should delete the file before loading. ● *Yes*: Indicates that the software should delete the file before loading. ● *No*: Indicates that the software should append to the existing file.	Target
Delimiters			
Column	Tab, Semicolon, Comma, Space, or any character sequence	For delimited file formats, this is the character sequence that indicates the end of one column and the beginning of the next.	New, Edit
Row	{new line}, {Windows new line}, {Unix new line}, or any character sequence	A character sequence that indicates where one row of data ends and the next begins.	New, Edit

Option	Possible values	Description	Mode
Row within text string	Character, Row delimiter	Defines how the row delimiter is interpreted within a text string. • *Character*: The specified row delimiter is treated as characters within the text string. • *Row delimiter*: The specified row delimiter is interpreted and defines rows within the text string.	New, Edit
Text	Single quotation marks ('), double quotation marks ("), or {none}	Denotes the start and end of a text string. All characters (including those specified as column delimiters) between the first and second occurrence of this character is a single text string. The treatment of the row characters is defined by the *Row within text string* setting.	New, Edit
Note Data in columns cannot include the column or row delimiter, unless you also specify a text delimiter. For example, if you specify a comma as your column delimiter, none of the data in the file can contain commas. However, if you specify a comma as the column delimiter and a single quote as the text delimiter, commas are allowed in strings in the data. You can use any ASCII characters (including non-printing characters) for column and row delimiters. You can specify an ASCII character by entering a forward slash (/) followed by the decimal representation of the character. For example, to use Y umlaut (Ÿ) as a delimiter, enter /255 in the delimiter property box.			
Default Format			
Escape char	Any character sequence, or {none}	A special character sequence that causes the software to ignore the normal column delimiter. Characters following the escape character sequence are never used as column delimiters. For example, suppose you specify a forward slash as the escape character and a comma as the column delimiter. Then, you must have a forward slash to have a comma appear inside a field.	New, Edit
NULL indicator	{none} or any other character sequence	Special character sequence that the software interprets as NULL data. The software ignores any NULL indicator specified in the file format for blob columns.	New, Edit
Ignore row marker(s)	{none} or any other character sequence	Character sequence, which when found at the beginning of rows, cause the software to ignore the row when reading the file or automatically creating metadata. To enter multiple character sequences, separate each with a semi-colon. To include a semi-colon or backslash as a marking character, precede with a backslash.	New, Edit, Source
Blank padding	leading, trailing	For fixed-width file format targets, pads extra blank spaces before or after the fields. • *Leading*: Adds blanks to the left of (before) the data.	New, Edit

Option	Possible values	Description	Mode
		- *Trailing*: Adds blanks to the right of (after) the data.	
Blank trimming	leading, trailing, both, or none	For fixed-width file format sources, trims extra blank spaces before or after the fields. - *Leading*: Trims blanks from the left of (before) the data. - *Trailing*: Trims blanks from the right of (after) the data.	New, Edit
Date	yyyy.mm.dd or other combinations	The date format for reading or writing date values to and from the file.	New, Edit
Time	hh24:mi:ss or other combinations	The time format for reading or writing time values to and from the file.	New, Edit
Date-Time	yyyy.mm.dd hh24:mi:ss or other combinations	The datetime format for reading or writing datetime values to and from the file.	New, Edit
Validate decimal data	Yes, No	For file targets, by default, the software converts data in delimited files to the decimal data type (even if it is in string form due to lazy decimal conversion) to make sure that the decimal format is valid. Note Lazy decimal conversion is an optimization whereby the data for columns of type DECIMAL is stored as STRING data type in rows and is converted to internal DECIMAL format only when they are used in an operation. To improve performance, you can manually deselect this conversion and validation operation. In the Target File Editor, set *Validate decimal data* to No.	Target
Input/Output			
Style	Headers or BOF/EOF	The format of the start and end of the file. Available properties in the Input/Output property group may change, based on this selection.	New, Edit
Skipped rows	Integer	For file formats using Headers style, the number of rows skipped when reading the file. Specify a non-zero value when the file includes comments or other non-data information.	New, Edit

Option	Possible values	Description	Mode
Skip row header	Yes, No	For file formats using Headers style, indicates whether the first row of data in the file contains the column names and should be skipped when reading the file. The software uses this property in addition to the *Skipped rows* property. When you select *Yes*, the software does not read data from the first row, and uses data in the first row to determine the file's column names.	New, Edit
Write row header	Yes, No	For file formats using Headers style, indicates whether to write column names in first row of output file.	New, Edit
Write BOM	Yes, No	For file formats using UTF-8 and UTF-16, determines the writing of BOM characters into the file. Choose Yes if you want to include BOM characters into a UTF-8 or UTF-16 file in which byte order is not otherwise defined. For a UTF-16 file, the software assumes the file to be UTF-16be, unless BOM characters are added by this property.	New, Edit
BOF Marker	Any character sequence, including a blank space, an empty string, or {none}	For file formats using BOF/EOF style, the string that marks the start of data in the file.	New, Edit
EOF Marker	Any character sequence, including a blank space, an empty string or {none}	For file formats using BOF/EOF style, the string that marks the end of data in the file.	New, Edit
Custom Transfer			
Program executable	File name	(Required) The name of the custom transfer program or its initialization script. For example: MyProgram.exe or MyProgram.cmd.	New, Edit
User name	Any character sequence, including a blank space, an empty string or {none}	(Optional) Log in ID for the server to which the custom transfer program connects. You may want to allow a custom program user to enter their user name when they enter their password in the software.	New, Edit
Password	Any character sequence, including a blank space,	(optional) Password for the server to which the custom transfer program connects. Passwords entered into this option are encrypted.	New, Edit

Option	Possible values	Description	Mode
	an empty string or {none}		
Arguments	Any character sequence, including a blank space, an empty string or {none}	(optional) You can create arguments in your custom transfer program and then specifically flag them from within the software using this box. For example, you might have security or compression mechanisms in your program. You can also link connection data to your transfer program's flags.	New, Edit
Locale	For more information about locales for sources, targets, and the software's internal processing, see *Locales and Multi-byte Functionality* [page 1793].		
Language	The three-letter language abbreviations specified in the ISO 639-2/T standard	Specifies the human language (for example, Korean, Japanese, or English) in which data is stored or processed. Select from the displayed list.	New, Edit
Territory	The two-letter territory abbreviations specified in the ISO 3166-1 standard.	Represents the geographical location (usually the country) where the language is used. The paring of a language and a territory determines factors such as date format, time format, decimal separator, and so on. For example, English is used differently in the United States and the United Kingdom.	New, Edit
Code Page	Shows the list of supported code pages. See Supported Locales and Encodings.	Specifies the sequence of bits that defines a character. For example, the Japanese code page contains ASCII, Greek, Cyrillic, and Japanese characters, thereby supporting the English, Greek, Russian, and Japanese languages.	New, Edit
Error Handling			
Log data conversion warnings	Yes, No	Determines whether to include data-type conversion warnings in the error log. Defaults to *Yes*.	New, Edit, Source
Log row format warnings	Yes, No	Determines whether to include row-format warnings in the error log. Defaults to *Yes*.	New, Edit, Source

Option	Possible values	Description	Mode
Log warnings	Yes, No	For unstructured file formats, determines whether to log warnings including: - When there are no files in the specified directory. - When no files match the specified filter. - When skipping an irregular file on UNIX (for example, a FIFO, symbolic link, character or block device, or UNIX socket). - When encountering an empty file with *Skip empty files* set to *Yes*.	New, Edit, Source
Maximum warnings to log	Integer greater than 0 or {no limit}	If *Log data conversion warnings* or *Log row format warnings* is enabled, you can limit how many warnings the software logs. Defaults to *{no limit}*.	New, Edit, Source
Capture data conversion errors	Yes, No	Determines whether to capture data-type conversion errors when processing a flat-file source. Defaults to *No*.	New, Edit, Source
Capture row format errors	Yes, No	Determines whether to capture row-format errors when processing a flat-file source. Defaults to *Yes*.	New, Edit, Source
Capture file access errors	Yes, No	For unstructured file formats, determines whether to log file-access errors when processing a flat-file source. Defaults to *Yes*.	New, Edit, Source
Capture string truncation error	Yes, No	Determines whether to capture string truncation errors when processing a flat-file source. Defaults to *No*.	New, Edit, Source
Maximum errors to stop job	Integer greater than 0 or {no limit}	If *Capture data conversion errors* or *Capture row format errors* is enabled, you can limit how many invalid rows the software processes before stopping the job. Defaults to *{no limit}*.	New, Edit, Source
Write error rows to file	Yes, No	Determines whether to write invalid rows to an error file. Defaults to *No*.	New, Edit, Source
Error file root directory	Directory path or blank	If *Write error rows to file* is enabled, type the root directory in which to save the error file. For added flexibility, you can enter a variable for this option. If you type a directory path here, then only enter the file name in the *Error file name* property. If you leave *Error file root directory* blank, then type a full path and file name in the *Error file name* property.	New, Edit, Source
Error file name	File name, file name including full path name, or blank	If *Write error rows to file* is enabled, type the name of the file in which to record the invalid rows. For added flexibility, you can enter a variable that is set to a particular file with full path name. Use variables to specify file names	New, Edit, Source

Option	Possible values	Description	Mode
		that you cannot otherwise enter such as file names that contain multi-byte characters.	
Source Information			
Include file name column	Yes, No	Determines whether to add a column that contains the source file name in the source output. Defaults to *No*. Change the value to *Yes* when you want to identify the source file in situations such as the following: ● You specified a wildcard character to read multiple source files at one time. ● You load from different source files on different runs.	Source
Column name		If the file name is included, the name of the column that holds the source file name. Defaults to *DI_FILENAME*.	Source
Column size		If the file name is included, the size (in characters) of the column that holds the source file name. Defaults to *100*. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.	Source
Include path		If the file name is included, determines whether to include the full path name of the source file. Defaults to *No*.	Source

The Column Attributes work area in the *File Format Editor* contains properties about the fields in the file format.

Property	Possible values	Description
Field name	Any sequence of letters or numbers, not including blank spaces	A name that identifies data in this column. If your file format uses the Headers style and you select *Yes* for the *Write row header* property, the software writes the field names in the target file.
Data type	blob, date, datetime, decimal, double, int, long, numeric, real, time, timestamp, varchar	The data type of values in this column. The long data type is not available in fixed-width formats.
Field size	Positive integer	If the data type is blob or varchar, specifies the number of characters in the field. For a blob column, the minimum field size is 1 and the maximum field size is 32768.
Precision	Positive integer	If the data type is decimal or numeric, specifies the total number of digits in the field.
Scale	Positive integer	If the data type is decimal or numeric, specifies the number of digits to the right of the decimal point.

Property	Possible values	Description
Format	{none}	For all data types other than varchar, specifies the format for this particular field. You can use this property to overwrite the default format. For example, if one date field is different than others, you can specify the different format here.
Content Type	{blank}, Address, Address_Primary_Name, Address_Primary_Number, Address_Primary_Postfix, Address_Primary_Prefix, Address_Primary_Type, Address_Secondary_Number, Country, Date, Delivery Point, DPV Status, Email, Family_Name1, Family_Name_Match_Std, Family_Name2, Family_Name2_Match_Std, Firm, Firm_Location, Firm_Location_Match_Std, Firm_Match_Std, Given_Name1, Given_Name1_Match_Std, Given_Name2, Given_Name2_Match_Std, Group_Number, Locality, Lot, Lot_Order, Name, Names_And_Firms, Phone, Postcode, Postcode1, Postcode2, Postname, Postname_Match_Std, Prename, Prename_Match_Std, Region, Sortcode_Rte, SSN, Title, Title_Match_Std	The name that specifies the type of data in a column. Typically use the field name, or a name similar to the field name. For example, if your field name is LastName, you may want to name the content type Family_Name.

If you delete a file format template from the object library, you must also delete all file sources and targets that are based on that file format template.

Related Information

Supported locales and encodings [page 1810]
HDFS file format [page 933]
Performance Optimization Guide: File multi-threading [page 2150]

3.2.2.12 Function

ƒ

Class

Reusable

Access

- For existing functions, click the *Functions* button in object editors.
- For imported functions, in the object library, click the *Datastores* tab, expand a datastore, and expand the *Functions* node.
- For custom or validation functions, click the *Custom Functions* tab in the object library or select *Tools* > *Custom Functions*.

Description

Use functions to process values. There are several types of functions:

- Built-in functions
- DBMS and application functions or stored procedures imported into SAP Data Services
- Custom functions you create
- Validation functions that you can import from SAP Information Steward or create locally

Functions have the following common attributes:

Attribute	Description
Name	The name of the function. This name appears in the function wizard and smart editor. It is also used when the function appears in a script or expression.
Description	Descriptive text entered when the function is created or imported into the software.
Function type	Each imported or custom function has a type. For imported and custom functions, right-click the function from the object library and select *Properties* to view the type. Descriptions and syntax for built-in functions is listed in the function wizard and the smart editor. Some funcitons also include a Category designation on the *Function* tab of the Properties dialog box.
Enable Parallel Execution	Check box on the Properties dialog box. Enables the software to run stored procedures and custom functions in parallel. This option must be selected in addition to entering a positive number for the parent data flow's degree of parallelism. For more information, see "Degree of parallelism" in the *Performance Optimization Guide*.
Validation function	Check box on the Properties dialog box indicating if the function is a validation function.

Related Information

Functions and Procedures [page 1511]

3.2.2.13 HDFS file format

Class

Reusable

Access

In the object library, click the *Formats* tab.

Description

An HDFS file format describes the structure of a Hadoop distributed file system. You can store templates for HDFS file formats in the object library. The format consists of multiple properties that you set in the editor. Available properties vary by the mode of the editor.

The HDFS file format editor includes most of the regular file format editor options plus the following options that are particular to HDFS. For a description of the modes and additional file format options, see *File format* [page 920].

After you add an HDFS file format to a data flow as a source or target, these options are also available in the source and target file editors.

Option	Possible values	Description	Mode
Data File(s)			
NameNode host	Computer name, fully qualified domain name, IP address, or variable	Name of the NameNode computer. If you use the default settings of *default* for NameNode host and *0* for NameNode port, the local Hadoop system uses what is set as the default file system in the Hadoop configuration files.	All
NameNode port	Positive integer or variable	Port on which the NameNode listens.	All
Hadoop user	Alphanumeric characters and underscores or variable	Hadoop user name.	All

Option	Possible values	Description	Mode
Pig			
Working directory	Directory path or variable	The pig script uses this directory to store intermediate data. If left blank, Data Services creates and uses the directory `/user/sapds_temp`, which is within HDFS.	All
Clean up working directory	Yes, No	Select *Yes* to delete the Pig output file and other intermediate files such as scripts and log files (including the `$LINK_DIR/log/hadoop` directory) after job execution. If *No* is selected, intermediate files remain in both the Pig Working Directory and the Data Services directory `$LINK_DIR/log/hadoop`.	All
Custom Pig script	Directory path or variable	Specify the location of a custom Pig script to run and use as an input for your job. A custom Pig script can contain any valid Pig Latin command, including calls to any MapReduce jobs that you want to use with Data Services. See the Pig documentation for information about Pig Latin commands. Custom Pig scripts must reside on and be runnable from the local file system of the machine that contains the Data Services Job Server configured for Hadoop (not on HDFS). Any external references or dependencies in the script should be available on the machine. To test your custom Pig script, you need to execute the script from the command prompt and check that it finishes without errors. For example, you could use the following: `$ pig -f myscript` In the Designer, you also need to do the following: 1. In the Format tab, right-click on HDFS Files and select *New*. The HDFS File Format Editor opens. 2. In the General section, set the *Type* option to Delimited. 3. In the Data File(s) section, set *Root directory* to the output location of the custom Pig script and *File name(s)* to the name of the files created (usually part*). 4. In the Pig section, set *Custom Pig script* to the path of the custom Pig script on the machine that contains the Data Services Job Server. 5. Define the schema of the output for the custom Pig script. 6. Set the delimiters to match the output. At this point, the file format can be used as a source. When you run the job, the custom Pig script also runs and you can then use the results for the rest of the job.	All

Option	Possible values	Description	Mode
Locale			
Code page	<default> us-ascii	For better performance, select *<default>* (UTF-8) or *us-ascii*, which are supported by Pig. For other code pages, Data Services uses HDFS API-based file reading.	All

Related Information

File format [page 920]
Hadoop [page 1841]

3.2.2.14 Log

Class

Single-use

Access

- To see the logs for jobs run on a particular Job Server, log in to the repository associated with the Job Server when you open the Designer. In the project area of the Designer, click the *Log* tab, and expand the job tree.
- To see the logs for jobs run on a particular Job Server, in the Administrator, select *Batch Jobs* > *Repository* (selecting the repository associated with the Job Server). Then, in the *Job Information* column for a job execution, click the type of log you want to view.

Description

A log records information about a particular execution of a single job.

- The *Log* tab in the Designer displays all logs for each execution. When you are finished with the logs for a given job or project, delete them from the *Log* tab. Right-click the log and select *Delete Log*.
- The *Job Information* column, of the Batch Job Status page in the Administrator also displays all logs for each execution.

There are three types of logs:

- Trace logs

- Monitor logs
- Error logs

Related Information

Trace logs [page 936]
Monitor logs [page 937]
Error logs [page 938]

3.2.2.14.1 Trace logs

The tracelog shows the execution progress through each component (object) of the job. It lists the process ID, thread ID, the object type being executed, the time each event began, and a description of the event.

For unsuccessful jobs, use the trace log to see which components of a partially executed job completed or where an error occurred.

Trace logs have the following information:

Entry	Description
Pid	Indicates the process identification number of the thread executing.
Tid	Indicates the thread identification number of the thread executing.
Type	Indicates the object being executed, such as a data flow or a transform. The generic job events are labeled TRACE. Possible types are listed and described in the following table.
TimeStamp	Indicates the date and time when the thread generated the message.
Message	Gives a description of the event that occurred as the thread was executing.

There are several types of traces.

Error number prefix	Description
ABAP	Traces the ABAP query execution.
ADMIN	Prints administrative information like "server not responding" or "power failure."
BLKLOAD	Traces bulk loading.
DATAFLOW	Traces the data flow execution.

Error number prefix	Description
EMAIL	Traces e-mail messages.
FTP	Traces FTP transport.
JOB	Traces the job execution.
OPTIMIZE	Records optimized details.
REPO	Traces objects in the repository.
ROW	Traces the row as it passes from one transform to another. It prints the row that is input to the transform and the output row it generates.
SQLFUNC	Traces function execution.
SQLLOAD	Traces loader execution, including the SQL sent to the target database.
SQLREAD	Traces reader execution, including the SQL sent to the source database.
SQLTRAN	Traces SQL transforms such as Table_Comparison and Key_Generation. The trace includes the SQL query sent to the underlying database and SQL results returned.
TRAN	Traces the transform execution.
USERFUNC	Traces user functions.

Related Information

Performance Optimization Guide: Checking system utilization [page 2108]

3.2.2.14.2 Monitor logs

The monitor log quantifies the activities of the components of the job. It lists the time spent in a given component of a job and the number of data rows which streamed through the component.

Use the monitor log to help tune the performance of a job.

Monitor logs have the following information:

Entry	Description
Path Name	Indicates which object is executing. The path name has the following format: `<dfname[_subdataflownumber]/objectname >`

Entry	Description

	where
	• **`<dfname>`** is the name of the data flow
	• **`<_subdataflownumber>`** is the number of the sub data flow if SAP Data Services split the data flow into multiple sub data flows
	• **`<objectname >`** is the name of the source, transform, or target that the data flow is processing
	For example, the following path name is for the first sub data flow of a data flow named Orders_DF, and the object being processed is a source named Order Details:
	`/Orders_DF_1/ORDER DETAILS`
	The next example is a path name for the first sub data flow of the Orders_DF data flow, and the object being processed is temporary storage for data from Order Details. The '`TS`' indicates that this object is temporary storage, and '`ORDERTEMP`' is the name of the temporary storage specified in the Data_Transfer transform.
	`/Orders_DF_1/TS_ORDER DETAILS_ORDERTEMP`
	The next example is a path name for the second sub data flow of the Orders_DF data flow, and the object being processed is a query transform:
	`/Orders_DF_2/Query`

State	Indicates the current status of the execution of the object. If you view the log while the job is running, this value changes as the status changes. The possible values are START, PROCEED, and STOP. In a successfully run job, all of these values are STOP to indicate that they finished successfully.
Row Count	Indicates the number of rows processed through this object. This value is updated based on the *Monitor sample rate (# of seconds)* set as a debug property.
Elapsed Time	Indicates the time (in seconds) since this object received its first row of data.
Absolute Time	Indicates the time (in seconds) since the execution of this entire data flow (including all of the transforms) began.

3.2.2.14.3 Error logs

The *error log* lists errors generated by SAP Data Services, by the source or target DBMS, or the operating system. If the error log is empty (that is, the button is dimmed), the job completed successfully.

Error logs have the following information:

Entry	Description
Pid	The process thread identification number of the thread executing.

Entry	Description
Tid	The thread identification number of the thread executing.
Number	An error number prefix (explained in the following table) and a number.
TimeStamp	The date and time when the thread generated the message.
Message	A description of the error that occurred as the thread was executing.

The error number prefixes are as follows:

Error number prefix	Description
ADM	Administration errors.
BAP	BAPI errors.
BIW	SAP BW errors.
CON	Connection errors. The connection indicated could not be initialized or failed during execution.
DBS	Database management system errors.
EML	Email errors.
FIL	Filespec errors.
OPT	Optimization errors.
PAR	Parser errors.
R3C	SAP connectivity errors.
R3S	SAP syntax errors.
REP	Repository errors.
RES	Resolver errors.
RUN	Runtime errors.
SCH	Job launcher errors.
SRV	Job Server errors.
SYS	System exceptions.
USR	User function errors.
VAL	Validator errors.
XRN	Transform errors.

3.2.2.15 Message function

ƒ

Class

Reusable

Access

In the object library, click the *Datastores* tab.

Description

Available in certain adapter datastores, message functions can accommodate XML messages when properly configured.

See your adapter's documentation for more specific information about the options available for a message function.

3.2.2.16 Outbound message

Class

Reusable

Access

In the object library, click the *Datastores* tab.

Description

Available in some adapter datastores, outbound messages are XML-based, hierarchical communications that SAP Data Services can publish to adapters. Outbound messages only wait for an acknowledgement from an external system; they do not wait for a reply. You can use outbound messages as targets only. You cannot use outbound messages as sources.

See your adapter's documentation for more specific information about the options available for outbound messages.

3.2.2.17 Project

Class

Single-use

Access

- Choose *Project* > *New*.
- In the object library, click the *Projects* tab.

Description

A project allows you to group jobs. It is the highest level of organization offered by SAP Data Services. Opening a project makes one group of jobs easily accessible in the user interface.

Projects have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the project area.

3.2.2.18 Query transform

Class

Single-use

Access

With a data flow diagram in the work space, click the Query transform icon in the tool palette, then click in the work space.

Description

A Query transform, like a SQL SELECT statement, retrieves a data set that satisfies the conditions you specify. With a Query transform, you can:

- Map columns from input to output schema
- Add new columns, nested schemas, and functions to the output schema
- Choose the data to extract
- Perform operations on the data
- Join data from multiple sources

A Query transform can operate on nested data. Using a Query transform, you can nest data or unnest nested data.

Query transforms have one attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

Related Information

Designer Guide: Query [page 310]

Designer Guide: Nested Data [page 342]

3.2.2.19 Real-time job

Class

Reusable

Access

- In the object library, click the *Jobs* tab.
- In the project area, right-click a project and select *Real-time Job*.

Description

A real-time job is a set of objects that you can execute together to process messages.

A real-time job is made up of three logical components:

- Initialization (optional)
- Real-time processing loop
- Clean-up (optional)

Each component can include the same objects as a batch job.

A real-time job is created in the Designer and then configured in the Administrator as a real-time service associated with an Access Server.

Start real-time services in the Administrator. If you have included any objects in the initialization component of a a real-time job, they run when the service starts. When a real-time service starts, a real-time processing loop registers itself and its message type with the Access Server and waits for the Access Server to send requests. The real-time processing loop continues to run until it encounters an error or you shut it down using the Administrator. The objects you placed inside the clean-up component of a real-time job run only when a service is shut down.

The message type that a given real-time job processes is determined (when it is designed) by the real-time source you include in the real-time processing loop; the format of the response is determined by the real-time target you include.

A real-time job has the same built-in attributes as a batch job:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the job.
Date created	The date when the object was created.

Like batch jobs, real-time jobs use the debug and trace properties to determine what information the software collects and logs when running the job. However, real-time jobs do not support the Enable Recovery debug options.

Related Information

Log [page 935]

3.2.2.19.1 Content of real-time job

A real-time processing loop can contain the following objects:

- A single data flow which can contain:
  - A single real-time source — XML message (required)
  - Sources — Files, XML files, and tables, including SAP tables
  - A single real-time target — XML message (required)
  - Targets — Files, XML files, tables, and template tables
  - Transforms, including queries
- Multiple data flows which can contain:
  - A single real-time source in the first data flow — XML message (required)
  - Sources — Files, XML files, and tables, including SAP tables
  - A single real-time target in the last data flow — XML message (required)
  - Targets — Files, XML files, tables, and template tables
  - Transforms, including queries
  - Memory tables — used to as staging tables to move data sets to the next data flow in the job.
- Multiple work flows, scripts, conditionals, while loops, etc.

Real-time jobs can also be built using IDocs. See the *Supplement for SAP* for more information.

> Note
>
> Real-time jobs can not use the following features: Data_Transfer transform, run as a separate process, or run at a job distribution level lower than a job level.

3.2.2.19.2 Arranging metadata

If you delete an object used in a real-time job from the object library, calls to the object are replaced with an icon indicating that the calls are no longer valid.

You can insert any number of work flows and data flows into a real-time job as long as the data flow models for a real-time processing loop are followed.

- A single data flow in a loop — Must have both a message source and target
- Multiple data flows in a loop — Must have a message source in the first data flow and a message target in the last data flow

At the job level, work flows and data flows cannot be designed to run in parallel. Inside a job level work flow, they can.

The messages in a data flow are significant. The following table indicates how data flows can be used.

If a data flow has:	It can be used as the:
One XML message source	First data flow in a real-time processing loop.
One XML message target	Last data flow in a real-time processing loop.
One XML message source and one XML message target.	Only data flow in a real-time processing loop.
No message source or target	Not the first, last, or only data flow in a real-time processing loop. Data flow in a batch job.

Data flows that do not contain messages can be used in batch jobs or in real-time processing loops (between its first and last data flows).

3.2.2.19.3 Message processing

Unlike batch jobs, real-time jobs are designed to process multiple messages rather than just files or tables of data.

For transforms that require all of the message's data at one time, such as queries that include aggregation functions, data is cached temporarily. The transform performs the specified operation, then clears caches in preparation for the next message.

To test a real-time job using the Designer, the recommended procedure is to test one message and create a target test file to receive the data. A real-time job will clear data after processing each message if system defaults are used. Therefore, deselect:

- *Delete data from table before loading* for a table target
- *Delete file* for a flat file target
- *Delete and re-create file* for an XML target

3.2.2.19.4 Loading targets as a single transaction

In a real-time job, you can load more than one table from a single datastore in a single transaction. When transactional loading is turned on for table targets, SAP Data Services sends `INSERT` statements for any of the tables included in the transaction to the database to process. You can also control which tables are loaded first by specifying the transaction order for the tables.

If the data flow includes a real-time target, it is always loaded in parallel with other targets to ensure load time is as short as possible.

3.2.2.19.5 Starting and stopping real-time services

For development and testing, you can manually execute a real-time job from the Designer. The Designer runs a real-time job in test mode.

When testing a real-time service or when running in production, the Access Server triggers the Job Server to process a request using the logic you built inside a real-time processing loop. The Access Server can also trigger the Job Server to shut down real-time processing loops. In a production environment, you control the operation of real-time services using the Administrator.

3.2.2.20 Script

Class

Single-use

Access

With a work flow or job diagram in the workspace, click the script icon in the tool palette.

Description

A script is a single-use object that assigns values to local, global or environment variables in a job or work flow. Define the script using the SAP Data Services scripting language.

Scripts have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

Related Information

Scripting Language [page 1709]

3.2.2.21 Source

Class

Single-use

Access

- To insert a document, table, or template table as a source, open the object library, go to the *Datastores* tab, select the object, drag it into the workspace, and select *Make Source*.
- To insert a flat file as a source, open the object library, go to the *Formats* tab, select the file format template for the file, drag it into the workspace, and select *Make Source*. Use the file format editor to specify the file's location.
- To insert an XML message or file as a source, open the object library, go to the *Formats* tab, select the an XML Schema or DTD format template, drag it into the workspace, and select *Make XML message source* or *Make XML file source*. Use the source editor to specify the test file name for the message or the source XML name for the file.
- To view options of a particular source, click the name of the source in the workspace or in the project area. This opens the appropriate editor, such as the table editor, or the XML file, XML message, or flat file format editors.

Description

A source is an object from which SAP Data Services reads data.

In a batch job, a source can be a document, a file, a table, a previously defined template table, an XML file, or a source-specific data flow (see your source-specific supplement for more options).

In a real-time job, a source can be a table, a previously defined template table, a flat file, an XML message, or an XML file. Each real-time job must have exactly one real-time data source.

You can make an embedded data flow a source.

Options available for sources from adapter datastores depend on the adapter implementation. Thus, options vary by data source and adapter version. See your adapter documentation for more information.

Related Information

Designer Guide: Embedded Data Flows [page 391]

3.2.2.21.1 Table source

You can tune performance by configuring the following common source options.

Option	Description
Make Port	Makes the source table an embedded data flow port.
Enable partitioning	Enables SAP Data Services to use the partition information in this table. If this option is selected, the software reads table data using the number of partitions in the table as the maximum number of parallel instances.
Join rank	Indicates the rank of the source relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before it joins sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For more information, see the Other Tuning Techniques section in the *Performance Optimization Guide*.
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. Options are: ● *Yes*: The source is always cached unless it is the outer-most source in a join. ● *No*: The source is never cached. The default is *Yes*. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. Best practice is to specify the cache only in the Query transform editor.
Array fetch size	Indicates the number of rows retrieved in a single request to a source database. The default value is 1000. Higher numbers reduce requests, lowering network traffic, and possibly improve performance. Maximum value is 5000. This option is available for source tables from DB2, Informix, ODBC, Oracle, and SQL Server datastores. When retrieving a column with an Oracle LONG data type, the software automatically sets *Array Fetch Size* to 1. If a column has an Oracle LONG data type, the software can only retrieve one row at a time.

For Oracle source tables you can use an overflow file for error handling. Select Yes for *Use overflow file* and enter a name for the file. Errors that occur while reading data are logged into the overflow file and the job execution proceeds while ignoring the rows that cause the error. To set the location of the overflow file directory use the table's Datastore Editor.

Related Information

Designer Guide: Creating embedded data flows [page 393]

Performance Guide: Using parallel execution [page 2137]

Performance Optimization Guide: Other tuning techniques, Join ordering [page 2197]

3.2.2.21.2 CDC table source

If a table comes from a CDC datastore, click the *CDC Options* tab and complete the following to set the Oracle, Attunity, SAP Sybase Replication Server and Microsoft SQL Server changed-data capture options. Options vary by source type.

Option	Description
CDC subscription name	The name used to mark sets of changed data read from the continuously growing CDC table. The subscription name marks the last row read so that the next job starts reading the CDC table from that position. You can use multiple subscription names to identify different users who read from the same imported CDC table. The subscription saves the position of each user. Select from the list or type a new name to create a new subscription. A subscription name must be unique within a datastore, owner, and table name. For example, you can use the same subscription name without conflict with different tables that have the same name in the same datastore if they have different owner names. This value is required for the Microsoft SQL server Change Tracking and Replication Server methods.
Enable check-point	Enables the software to restrict CDC reads using check-points. Once a check-point is placed, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last check-point. By default, check-points are not enabled.
Automatically delete rows after reading	For Microsoft SQL Server only. If the option is cleared, no rows are deleted from the CDC table. If the option is selected, the behavior is different depending on the CDC method used: ● Replication Server method: If more than one data flow uses the same CDC table as a source, only the rows that have been read by all readers are deleted. If any of those readers have the option cleared, no rows are deleted. ● Changed-data capture (CDC) method: All rows of the CDC table are deleted regardless of whether other readers have finished reading from the table. Therefore, if more than one data flow uses the same CDC table as a source, only select this option for the data flow that executes last.

Option	Description
Get before-image for each update row	Some databases allow two images to be associated with an UPDATE row: a before-image and an after-image. If your source can use before-images and you want to read them during change-data capture jobs, enable this option. By default, only after-images are retrieved.

Related Information

Designer Guide: Capturing Changed Data [page 738]

3.2.2.21.3 Flat file source

A flat file source has the *Join rank* and *Cache* options in addition to the file format options. For these two options, SAP Data Services uses the same interpretation for both files and tables.

The flat file source options are the same in the source editor as in the new object editor with the following exception. For the Unstructured text and Unstructured binary file format types, there is an additional option:

Number of files to read: Indicates the maximum number of files to read. A value of zero reads all files. The default is blank.

Related Information

Table source [page 948]
File format [page 920]

3.2.2.21.4 Hadoop Hive Adapter Source

You can set the following options on the *Adapter Source* tab of the source table editor.

Option	Possible values	Description
Clean up working directory	True, False	Select *True* to delete the working directory after the job completes successfully.
Parallel process threads	Positive integers	Specify the number of threads for parallel processing, which can improve performance by maximizing CPU usage on the Job Server computer. For

Option	Possible values	Description
		example, if you have four CPUs, enter 4 for this option.

Related Information

Performance Optimization Guide: Using Parallel Execution, File multi-threading [page 2150]

3.2.2.21.5 Persistent cache source

A persistent cache source has the following options:

Option	Description
Make Port	Makes the persistent cache source an embedded data flow port.
Join rank	For this option, SAP Data Services uses the same interpretation for both persistent cache tables and database tables.
Cache	For this option, the software uses the same interpretation for both persistent cache tables and database tables.
Table name	The *Table name* box displays the name that you entered when you created the persistent cache table. You cannot change this name.
Table owner	The *Table owner* box displays the owner that you entered when you created the persistent cache table. You cannot change this name.
Datastore name	The *Datastore name* box displays the name that you entered when you created the persistent cache. You cannot change this name.
Database type	The *Database type* box displays the *Persistent Cache* option, which you cannot change.

Related Information

Table source [page 948]
Designer Guide: Creating embedded data flows [page 393]

3.2.2.21.6 SAP sources

SAP sources include Open Hub tables and SAP ODP sources.

For details on SAP sources, refer to the *SAP Data Services Supplement for SAP*.

Related Information

Supplement for SAP: Open Hub Table source [page 2565]
Supplement for SAP: ODP source [page 2557]

3.2.2.21.7 Teradata source

The *Teradata options* tab for a Teradata source includes the following modes.

- Parallel transporter API
- Parallel Transporter Export Operator
- None

The availablity of *Advanced* options differs between modes. If an option is empty in the *Advanced* section, Data Services uses the default value specified at the database level.

For details on the following options, refer to your Teradata documentation.

Option	Description
General	
Clean up bulk reader directory after export	Select *Yes* to delete all files in the bulk reader directory after successfully exporting. Select *No* to leave the files in the directory.
Minimum number of sessions	Specifies the minimum number of sessions required for the Export driver job to continue. Default is one session.
Maximum number of sessions	Specifies the maximum number of connections to Teradata. Must be greater than zero. Defaults to one session per available AMP. Use this parameter in conjunction with *Number of export operator instances* and *Parallel process threads* for performance tuning when reading from a Teradata source. For large volumes of data, more sessions allows more data to be read in parallel. Ideally this number should equal the number of AMPs.
Number of export operator instances	Specifies the number of instances for export operators. Use this parameter in conjunction with *Maximum number of sessions* and *Parallel process threads* for performance tuning when reading from a Teradata source. Multiple export instances can improve performance. Ideally this value should equal the number of CPUs.
Tenacity hours	Specifies the number of hours the Export driver attempts to log on when the maximum number of load and export operations are already running on the Teradata database. Default is 4 hours.

Option	Description
Tenacity sleep	Specifies the number of minutes the Export driver pauses before attempting to log on when the maximum number of load and export operations are already running on the Teradata database. Default is 6 minutes.
Data handling	
Block size	Specifies the block size (in bytes) when returning data to the client.
Data encryption	Select *Yes* to enable full security encryption of SQL requests, responses, and data. Select *No* to disable encryption.
Query band session	Specifies a user-defined query band expression to be set for every SQL session.
Notification	
Level	Indicates the level at which certain events are reported: ● Off: No notifications. Default. ● Low: Notifications occur for Initialize, CLIv2/DBS Error, Exit. ● Medium: Notifications occur for all events except File or OUTMODE Open and Statement Fetch Begin and End. ● High: Notifications occur for all events.
Method	Specifies the method for reporting events: ● None: No event logging. Default. ● Message: Send events to a log (for example the EventLog on Windows). ● Exit: Send the events to a user-defined notify exit routine and to the system log.
User-defined string	Specifies a user-defined string that precedes all messages sent to the system log.
User-defined exit routine	Specifies the name of a user-defined notify exit routine.
Trace	
Level	Specifies the type(s) of diagnostic messages each instance of the driver wirtes to a log file. API mode writes to external log files and Export Operator mode writes to public or private logs. ● CLI: Activates the tracing function for CLIv2-related activities. ● PX: Activates the tracing function for activities involving the common library. ● Oper: Activates the tracing function for driver-specific activities. ● Notify: Activates the tracing function for activities related to the Notify feature.
Tracing file	For API mode, specifies the name of the external log file used for trace messages.

Option	Description
Miscellaneous	
Parallel process threads	Specifies the number of threads for parallel processing, which can improve performance by maximizing CPU usage on the Job Server computer. Use this parameter in conjunction with *Maximum number of sessions* and *Number of export operator instances* performance tuning when reading from a Teradata source. The data loads into buffers in Data Services, and the parallel process threads break these buffers into rows and columns. Ideally this number should equal to the number of CPUs.
Logon mechanism	Specifies which logon mechanism to use: ● Kerberos 5 ● NT Lan Manager ● Lightweight Directory Access Protocol ● Simple and Protected GSSAPI Negotiation Mechanism
Logon mechanism data	Specifies additional optional logon mechanism data.
AccountId	An optional attribute that specifies the account associated with the user name (the user specified in the datastore).
Private log name	Specify the name of a log that is maintained by the Teradata Parallel Transporter Logger inside the public log. The private log contains all of the output provided by the Export operator.

Related Information

Performance Optimization Guide: Bulk Loading and Reading, Bulk loading and reading in Teradata [page 2178]
Performance Optimization Guide: Using Parallel Execution, File multi-threading [page 2150]
Teradata [page 902]
Teradata target table options [page 989]

3.2.2.21.8 XML file source

An XML file source has the same *Join rank* and *Make port* options as tables.

An XML file source has the following options in addition to its read-only XML Schema or DTD format information:

Option	Description
XML file	The location relative to the Job Server of an XML-formatted file to use as the source. You can enter a variable for this option.

Option	Description
	Note If your Job Server is on a different computer that the Designer, you cannot use Browse to specify the file path. You must type the path. You can type an absolute path or a relative path, but the Job Server must be able to access it.
Enable validation	A check box to turn on the comparison of the incoming data to the stored XML Schema or DTD format. When this option is enabled, the data flow throws an exception if the incoming source is not valid.
Include file name column	Determines whether to include a column in the source output that contains the source XML file name. Defaults to *No*. Change the value to *Yes* when you want to identify the source XML file in situations such as the following: ● You specified a wildcard character to read multiple source XML files at one time ● You load from different source XML files on different runs
Modify	If the file name is included, this button enables you to modify *File name column* and *Column size*.
File name column	If the file name is included, the name of the column that holds the source file name. Defaults to *DI_FILENAME*.
Column size	If the file name is included, the size (in characters) of the column that holds the source file name. Defaults to *100*. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.
Include path	If the file name is included, determines whether to include the full path name of the source file. Defaults to *No*.

Related Information

Table source [page 948]

3.2.2.21.9 XML message source

An XML message source has the same *Make port* option as tables.

The XML message source has these options in addition to its read-only XML Schema or DTD format information:

Option	Description
XML test file	The location relative to the Job Server of an XML-formatted file to use as the message source when you execute the job in test mode. Note If your Job Server is on a different computer that the Designer, you cannot use Browse to specify the file path. You must type the path. You can type an absolute path or a relative path, but the Job Server must be able to access it. A variable can also be used.
Enable validation	A check box to turn on the comparison of the incoming message to the stored XML Schema or DTD format. When this option is selected, the real-time job throws an exception if the incoming message is not valid. When you are developing a real-time job, this validation helps you to make sure sample data is both valid and well-formed. If you select this option in production, make sure to include appropriate error handling either in the SAP Data Services job or the client application to process an error caused if a data flow in the real-time job receives data that does not validate against the imported format.
`<implied>` : Join rank	The XML message source is always the outer table in a join. You cannot change its join rank. This option is implied and does not appear in the editor.

Related Information

Table source [page 948]

3.2.2.22 Table

Class

Reusable

Access

In the object library, click the *Datastores* tab. Expand a datastore to find the tables node. Expand this node to view the list of imported tables. Right-click and select *Properties* to view and edit table properties.

Description

You can use a table as a source or target in a data flow.

The *Indexes* tab on the Properties window for a table shows information about the table's indices. Under *Index*, the window lists the primary index followed by any secondary index. Select an index and the window lists the columns in that index under *Column*.

The *Partition* tab on the Properties window displays how table metadata is partitioned. Partitions can be imported with a table or you can create metadata for them within SAP Data Services.

The *Attributes* tab on the Properties window displays built-in table attributes:

Table Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	A configurable description field.
Business_Name	Supports metadata exchanged between Data Services and SAP BusinessObjects Universe Builder (UB) 1.1. A configurable field for the logical Business_Name used in SAP BusinessObjects Universe Builder. Data Services can extract, transform and load data while transferring this information intact.
Business_Description	Supports metadata exchanged between Data Services and SAP BusinessObjects Universe Builder (UB) 1.1. A configurable, business-level description of the table. Data Services transfers this information separately and adds it to any SAP BusinessObjects Universe Builder Class description.
Table_Usage	A configurable label field. Use it to mark a table as `fact` or `dimension` for example.
Total_Number_Of_ Rows_Processed	The number of rows loaded into the table in the last successful load.
Date_last_loaded	The time the table was last successfully loaded.
Number_Of_Rows_ Rejected	The number of rows rejected in the last successful load.
Number_Of_Inserts	The number of rows inserted in the last successful load.
Number_Of_Updates	The number of rows updated in the last successful load.

Table Attribute	Description
Date_Created	The date that the object was created.
Estimated_Row_ Count	A configurable estimate of the table size used in calculating the order in which tables are read to perform join operations. Used for SAP tables only.
Number_Of_Deletes	The number of rows deleted in the last successful load.
Elapsed_Time_For_ Load	The time it took to load this table in the last successful load.
Table_Type	The type of datastore object for tables and hierarchies. Most often the value `TABLE` is displayed. However, the software might display the following values for SAP sources: BW master data transfer, BW transaction data transfer, BW hierarchy data transfer, SAP hierarchy.
SAP_Table_Class_ Name	Imported with SAP table metadata.
Loader_Is_Template_Table	If `YES`, indicates that the table is an internal, template table created in the software. Before running production jobs, execute the job to load the target table then right-click the template table in the object library or in a data flow and the software creates the table in your database and imports it.
SavedAfterCheckOut	If `YES`, indicates you saved the table after it was checked out of the central repository. The software uses this information to determine whether to save the table in the central repository when it is checked in.
PartitionModified	If `YES`, indicates that you modified the partitions in this table using the software after you imported the table's metadata.

Related Information

Performance Optimization Guide: Overview of parallel execution [page 2137]
Template table [page 996]

3.2.2.22.1 Column attributes for tables

SAP Data Services also supports column attributes for tables.

3.2.2.22.2 To view column attributes for a table

1. In the *Datastores* tab of the object library, double-click a table.

   The Table Metadata window opens.
2. Right-click a column name and select *Properties*.

   The Column Attributes window opens.
3. Click the *Attributes* tab.

Column Attribute	Description
Business Name	A configurable logical name.
Business_ Description	A configurable business-level description of the column.
Column_Usage	Supports metadata exchanged between SAP Data Services and SAP BusinessObjects Universe Builder (UB) 1.1. Enter `Dimension`, `Measure`, or `Detail` as a value for a corresponding object in SAP BusinessObjects Universe Builder. ● If set to `Dimension`, the corresponding object created in SAP BusinessObjects Universe Builder is of qualification type *Dimension*. ● If set to `Measure`, the corresponding object is of qualification type *Measure*. ● If set to `Detail`, the corresponding object is of qualification type *Detail* and requires you to set a value for *Associated_Dimension*.
Associated_ Dimension	Set this value only if *Column_Usage* is set to `Detail`. The value must be in the format: `table.column`. The Detail column is created under the Dimension column you specify.
Acta_autojoin	Generated by SAP Data Services. Not configurable.
Associated_ domain	Use for databases that use domains such as PeopleSoft.
Physical_Name	Use for applications that allow logical names for a column such as Oracle Applications.

The attributes listed above are available for all tables.

Related Information

Designer Guide: Attributes that support metadata exchange [page 735]
Attributes for DTDs [page 908]
Attributes supported for XML schemas [page 1010]

3.2.2.23 Target

Class

Single-use

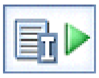
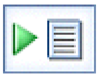
Access

- To display target options, click the name of the target in the workspace or in the project area. This opens the object editor.
- To display target properties, right-click a target and choose *Properties*.

Description

A target is an object to which SAP Data Services loads extracted and transformed data in a data flow.

In a data flow, a target can be one of the objects that the following table shows.

Icon for target object	Target object description
	Document
	Flat file
	Outbound message
	Table
	Template table
	XML file
	XML message
	XML template

You can make a target an embedded data flow port: set the *Make port* option to *Yes* for flat files; select the *Make port* check box for other targets.

This section describes:

- Target files
- Target persistent cache tables
- Target tables
- Target Data_Transfer files and tables
- Target XML files, messages, and templates

Documents and outbound messages are only available from adapter datastores. Options available for these targets depend on the adapter implementation. Thus, options vary by data source and adapter version. See your adapter documentation for more information

Related Information

Designer Guide: Embedded Data Flows [page 391]

Designer Guide: Creating embedded data flows [page 393]

3.2.2.23.1 Target files

You can use any flat file format as a target in a data flow. To add a target file, select a file format in the object library, drag the file format into the data flow workspace, and select *Make Target*.

If the schema defined in the file format does not match the schema that is input to the target, SAP Data Services provides validation errors to identify the mismatch.

Use the file format editor in target mode to edit the file format of a target file. You cannot edit all properties of a particular target file. You can change some properties of the file format.

You can also change the name of the target file object using the object's properties. Right-click the object and choose *Properties*.

Related Information

File format [page 920]

3.2.2.23.2 Target persistent cache tables

To create a new persistent cache target table, take one of the following actions:

- Select a template table under a persistent cache datastore in the object library, drag the table into the workspace, and type a name for the table.
- Click the template table icon in the tool palette, click the workspace, choose a persistent cache datastore, and type a name for the table.

i Note

You cannot update a persistent cache table. If the data within it changes, you must recreate it and load it.

Table 107: Target

Option	Description
Make port	Select this check box to make the target table an embedded data flow port.
Table name	The *Table name* box displays the name that you entered when you created the persistent cache table. You cannot change this name.
Table owner	The *Table owner* box displays the owner that you entered when you created the persistent cache table. You cannot change this name.
Datastore name	The *Datastore name* box displays the name that you entered when you created the persistent cache. You cannot change this name.
Database type	The *Database type* box displays `Persistent_Cache` which you cannot change.

Table 108: Options

Option	Description
Column comparison	Specifies how the input columns are mapped to persistent cache table columns. There are two options: ● *compare_by_position*: SAP Data Services disregards the column names and maps source columns to target columns by position ● *compare_by_name*: the software maps source columns to target columns by name
Include duplicate keys	Specifies whether or not to include duplicate keys in the persistent cache. Defaults to selected.

Table 109: Keys

Option	Description
Key column	Specify one or more columns to use as the key for the persistent cache. Click the arrow to view a drop-down list of column names. To change the order of the columns in the key, use one of the following options: ● Right-click the column and select *Move Up* or *Move Down*. ● Select the column and click the down or up arrow in the top right corner of the *Keys* tab. To remove a column, use one of the following options: ● Right-click the column and select *Delete*. ● Select the column and click the delete icon in the top right corner of the *Keys* tab.

Related Information

Designer Guide: Embedded Data Flows [page 391]

3.2.2.23.3 Target tables

You can add a table to a data flow diagram as a target if SAP Data Services can write to the application or database containing the table. To add a target table, select the table in the object library, drag the table into the workspace, and select *Make Target*.

If the schema defined in the table does not match the schema that is input to the target, the software provides validation errors to identify the mismatch.

When loading DB2, ODBC, or Oracle tables, the software parameterizes the SQL. Parameterized SQL statements result in quicker load times. To parameterize SQL, the software must be able to generate, parse, and compile the statement. For example, the software is unable to parameterize SQL when using transactional loading or triggers.

You configure a target table by setting options in the target editor. Available options depend on the database in which the table is defined.

3.2.2.23.3.1 Common target table options

This table describes the options common to all supported database types.

Table 110: Target tab

Option	Description
Make port	Select this check box to make the target table an embedded data flow port.
Database type	Select an item in the *Database type* box to set the content of additional tabs on the target table editor to match the specific options for that database type. This option allows you to quickly set target option values in data flows. If your target datastore has multiple configurations, all database types and their version numbers, which you specified in these configurations, are listed. To add or remove items in this list, edit the datastore configuration information in the datastore editor. The software allows you to use target table editor option values from any datastore configuration: ● If the datastore has only one configuration, then the initial values for the target table editor are defaults set by Designer for that database type or version. ● If the datastore has more than one configuration and there are different database types/versions, then the software determines the initial values for the additional database types/versions from the *Use values from* box in the Create New Configuration dialog (a sub-dialog of the datastore editor). ● If you also select the *Restore values if they already exist* check box (in the Create New Configuration dialog), SAP Data Services looks for previously defined values that once existed for that database type or version. It is possible for a data flow to contain target table editor values for a database type or version, even if its datastore configuration was deleted. The software retains all target table editor values saved with every datastore configuration. If such values exist, then it restores those values. Otherwise, it gets the values from the configu-

Option	Description
	ration you select from the *Use values from* option. For example, suppose you set a configuration for Oracle 8i. When you edit the target table editor options, you change the *Rows Per Commit* default value of 1000 to 500. Later you add a new datastore configuration for a Microsoft SQL Server 2000 database to your original datastore and set the *Use values from* option to Oracle 8i. In this case, the target table editor settings for SQL Server inherit the value 500 for *Rows per Commit* because this was the value set in the Oracle 8i configuration. The values you set for the options in the target table editor are specific to the instance and database type/version of that object in the data flow. If you set values for one target table, any other target table in the same data flow is not affected.

Table 111: Options tab

Option	Description
Rows per commit	Specifies the transaction size in number of rows. For example, if set to 1000, the software sends a commit to the underlying database every 1000 rows. Load triggers are never split across transaction boundaries, so if the load trigger crosses transaction boundaries, then the size of the transaction is automatically extended to accommodate the entire trigger requirement. For example, suppose you set *Rows per commit* to 3 and specified an insert trigger, where an incoming insert statement is converted into 5 statements. *Rows per commit* would automatically be extended to 5 to accommodate each insert trigger statement in a single transaction. This option is not available for targets in real time jobs.
Delete data from table before loading	For batch jobs, to clear the contents of the table before loading it, sends a TRUNCATE statement to databases that support it (Oracle, Microsoft SQL Server, SAP Sybase) or sends a DELETE statement to databases that do not support TRUNCATE. Default setting is not selected. For real-time jobs, clears data after processing each message. For real-time jobs you may want to deselect this option during development and testing.
Drop and re-create table	Drops the existing table and creates a new one with the same name before loading. This option displays only for template tables. Template tables are used in design or test environments.
General	
Column comparison	Specifies how the input columns are mapped to output columns. There are two options: *Compare by position*: The software disregards the column names and maps source columns to target columns by position. *Compare by name*: The software maps source columns to target columns by name. Validation errors occur if the data types of the columns do not match.

Option	Description
Number of loaders	Loading with one loader is known as “single loader loading”. Loading when the number of loaders is greater than one is known as “parallel loading”. The default number of loaders is 1. You can specify any number of loaders. When parallel loading, each loader receives the number of rows indicated in the Rows per commit option, in turn, and applies the rows in parallel with the other loaders. For example, if you choose a *Rows per commit* of 1000 and set the *Number of Loaders* to 3, the first 1000 rows are sent to the first loader. The second 1000 rows are sent to the second loader, the third 1000 rows to the third loader, and the next 1000 rows back to the first loader.
Enable Partitioning	(Displayed only if the target table data is partitioned) Loads data using the number of partitions in the table as the maximum number of parallel instances. You can only select one of the following loader options: ● Number of Loaders ● Enable Partitioning ● Transactional Loading Note If you set *Enable Partitioning* to Yes and *Include in transaction* to Yes, the *Include in transaction* setting overrides the *Enable Partitioning* option.
Error handling	
Use overflow file	This option is used for recovery purposes. If a row cannot be loaded it is written to a file. When this option is set to Yes, options are enabled for the file name and file format. The default setting is No.
File name *File format*	These options are available only when you select Yes for the *Use overflow* file option. Specifies the file name and file format for the overflow file. The overflow format can include the data rejected and the operation being performed (write_data) or the SQL command used to produce the rejected operation (write_sql). You can enter a variable for the file name.
Update control	
Use input keys	If the target table contains no primary key, this option enables the software to use the primary keys from the input. The default setting is No.
Update key columns	This option is set to No by default. If you select Yes for this option, the software updates key column values when it loads data to the target.
Auto correct load	Select Yes to use auto correct loading. Auto correct loading ensures that the same row is not duplicated in a target table. This is particularly useful for data recovery operations. The default setting is No.

Option	Description
	Note This option is not available for targets in real time jobs or target tables that contain LONG column(s). When you select Yes for this option, the software reads a row from the source, then checks if the row exists in the target table with the same values in the primary key. If *Use input keys* is set to Yes, the software uses the primary key of the source table. Otherwise, the software uses the primary key of the target table; if the target table has no primary key, the software considers the primary key to be all the columns in the target. If a matching row does not exist, a new row is inserted, regardless of other options. If a matching row exists, the row is updated depending on the values of *Ignore columns with value*, and *Ignore columns with null*: ● When the column data from the source matches the value in *Ignore columns with value*, the corresponding column in the target table is not updated. The value may be spaces. Otherwise, the corresponding column in the target is updated with the source data. ● When the *Ignore columns with null* option is set to Yes and the column data from the source is NULL, then the corresponding column in the target table is not updated. Otherwise, the corresponding target column is updated as NULL since the source column is NULL. For supported databases, when the *Allow merge or upsert* option is enabled, the software can optimize data flows such that the database completes the auto correct load operation. When all other operations in the data flow can be pushed down to the source database, the auto-correct loading operation is also pushed down. The generated SQL implements the *Ignore columns with value* value if completed in the target editor, and the *Ignore columns with null* Yes/No setting.
Allow merge or upsert (This is available for Microsoft SQL Server 2008 and higher only)	Specifies whether the Optimizer may use a MERGE statement to improve the performance of auto correct load functionality. *Yes:* Allows the Optimizer to consider using a MERGE statement during an auto correct load operation. If the Optimizer does not use a MERGE statement, it uses a T-SQL block to identify, insert, and update rows. *No:* The Optimizer will not use a MERGE statement to improve auto correct load performance. The default is Yes. **Note** When the data flow contains a Query transform with an Order by clause, the Optimizer always uses a T-SQL block to identify, insert, and update rows.

Option	Description
Ignore columns with value	Enter a value that might appear in a source column and that you do not want updated in the target table. When this value appears in the source column, the corresponding target column is not updated during auto correct loading. You can enter spaces.
Ignore columns with null	Select Yes if you do not want NULL source columns updated in the target table during auto correct loading. This option is only available when you select Yes for the *Auto correct load* option.
Transaction control	
Include in transaction	Indicates that this target is included in the transaction processed by a batch or real-time job. This option allows you to commit data to multiple tables as part of the same transaction. If loading fails for any one of the tables, no data is committed to any of the tables. Transactional loading can require rows to be buffered to ensure the correct load order. If the data being buffered is larger than the virtual memory available, the software reports a memory error. The tables must be from the same datastore. If you choose to enable transactional loading, other options are not available: ● Rows per commit ● Use overflow file, and overflow file specification ● Number of loaders ● Enable partitioning ● Bulk loader options ● Pre load commands ● Post load commands ● Delete data from table before loading The software does not push down a complete operation to the database if transactional loading is enabled.
Transaction order	Transaction order indicates where this table falls in the loading order of the tables being loaded. By default, there is no ordering. All loaders have a transaction order of zero. If you specify orders among the tables, the loading operations are applied according to the order. Tables with the same transaction order are loaded together. Tables with a transaction order of zero are loaded at the discretion of the data flow process. In Data flow view, the specified transaction order number appears nearer to the corresponding target.
Commit at end of INSERT... SELECT	When set to Yes (default), the software commits a single transaction log statement per load. When set to No, transaction size is limited to the value specified in *Rows per commit* to create a smaller transaction size than the single commit per job default behavior.

Option	Description
	If your transaction log size is too small for a single transaction of this type, set a commit size when the following applies: • the job has a source and target that use the same datastore • the job has an Oracle target table • the software is optimizing and executing the job by pushing down the read operation to the Oracle target table host. This type of operation requires that the software generate an INSERT...SELECT SQL statement. The software commits a transaction for an INSERT...SELECT by default at the end of the job. • the job failed with an Oracle transaction log full error. Troubleshooting: 1. View the SQL that the software generates to see if an INSERT...SELECT statement is in use. (In the Designer, open a data flow and select ▸ *Validate* ▸ *Display Optimized SQL* ◂.) 2. If so, set the Commit at end of INSERT...SELECT to No and enter a value for *Rows per commit*. If you use this option, expect to see a decrease in performance.
Use NVARCHAR for VARCHAR columns in supported databases	The software creates **`<nvarchar>`** columns in the template table for all **`<varchar>`** columns in the input schema of the data flow. The data type displays as varchar in the Designer, and, when supported by the DBMS, as nvarchar in the database table. The following database management systems do not support the **`<nvarchar>`** data type: • DB2 (non-UTF-8) • Oracle 8.x • Informix • SAP Sybase ASE • SAP Sybase IQ For these DBMSs, the software creates columns with varchar data types and increases the column size using a codepage conversion factor based on the client code page defined in the datastore. ⚠ Caution Data loss may occur when transcoding from one national language to a different national language. Data loss will not occur when transcoding from a national language to Unicode. ⚠ Caution Data truncation occurs when the column size of the source exceeds the maximum size allowed by the target DBMS.

Load triggers tab

Specifies SQL commands performed by the database on an INSERT, UPDATE, or DELETE operation.

You can specify a **`<load trigger >`** (a template SQL statement) that has placeholders for column and variable values. The software sets the placeholders at execution time based on the fields in the transform's input schema. For each row, the template is filled out and applied against the target.

The special operations you specify in a load trigger can occur before, after, or instead of normal operations.

Use load triggers in situations such as archiving updates to a warehouse or incremental updates of aggregation value.

The software does not parse load triggers. Thus, when you specify a load trigger, the software does not parameterize SQL statements. As a result, load times might be higher when you use load triggers.

The software does not validate load triggers.

> Note
>
> If you use an override, you cannot specify auto correct load.

For example, instead of applying an insert of a new sales order rows, you use a load trigger that applies inserts and updates of aggregated values of sales_per_customer and sales_per_region.

The templates give you a row with customer_id, order_amount, region_id, and so forth.

The INSERT and UPDATE statements are:

```
INSERT into order_fact
values ([customer_id], [order_amount]);
UPDATE region_fact
SET order_amount =
order_amount + [order_amount]
WHERE region_id = [region_id];
```

Enter your load triggers manually or drag column names from the input schema. Column names must be enclosed in curly braces or square brackets. For example, `{SalesOffice} or [SalesOffice]`.

With curly braces, the software encloses the value in quotation marks, if needed. With square brackets, it will not. To avoid unintended results, use curly braces for varchar or char column names.

If you insert column names into the SQL statement by dragging the column names, the software inserts square brackets for you. If you require curly braces, you must make the change from square brackets to curly braces.

`[#insert]`, `[#update]`, and `[#delete]` represent the default operations.

To delimit a SQL statement, use `[#new]`. For example:

```
[#insert] [#new]
insert into foo values ([col1], {col2}, ...)
```

For UPDATE operations you must specify both the "before" and the "after" image values. You can specify both images for INSERT and DELETE operations, also, but it is not required.

To specify "before" images, add the suffix `.before` to the column name. To specify "after" images, add the suffix `.after` to the column name.

The default suffix for UPDATE and INSERT operations is `.after`. The default suffix for DELETE operations is `.before`.

You can include variables in the SQL statements, but not expressions.

You can map a batch of SQL statements. Each SQL statement is separated by a new separator (`[#new]`).

The following statement is an example for mapping insert SQL:

```
INSERT into log_table values ({col1}, {col2})
[#new]
[#insert] [#new]
delete from alt_junk where . . .
```

Pre Load Commands tab and *Post Load Commands tab*

Specify SQL commands that the software executes before starting a load or after finishing a load.

When a data flow is called, the software opens all the objects (queries, transforms, sources, and targets) in the data flow. Next, the software runs the target's preload script. Therefore, the software executes any preload SQL commands before processing any transform.

> i Note
>
> Because the software executes the SQL commands as a unit of transaction, you should not include transaction commands in preload or postload SQL statements.

Both the *Pre Load Commands* tab and the *Post Load Commands* tab contain a *SQL Commands* box and a *Value* box. The *SQL Commands* box contains command lines. When you first open the tab, an empty line appears.

To edit a line, select the line in the *SQL Commands* box. The text for the SQL command appears in the *Value* box. Edit the text in that box.

To add a new line, determine the desired position for the new line, select the existing line immediately before or after the desired position, right-click, and choose *Insert Before* to insert a new line before the selected line, or choose *Insert After* to insert a new line after the selected line. Finally, type the SQL command in the *Value* box.

To delete a line, select the line in the *SQL Commands* box, right click, and choose *Delete*.

You can include variables and parameters in preload or postload SQL statements. Put the variables and parameters in either brackets, braces, or quotes. The software translates each statement differently, writing a statement that depends on the variable or parameter type.

Entered statement	Variable value	Written statement
`[$X]`	5	`5`
`[$X]`	John Smith	`John Smith`
`{$X}`	5	`5`
`{$X}`	John Smith	`'John Smith'`
`'$X'`	5	`'$X'`
`'$X'`	John Smith	`'John Smith'`

You cannot use Pre Load and Post Load SQL commands in a real-time job.

Related Information

Designer Guide: Embedded data flows [page 392]

Message processing [page 945]

3.2.2.23.3.2 DB2 target table options

The following table contains option and description information specific to DB2 target tables.

Table 112: Target tab

Option	Description
Make port	Specifies the table as an embedded data flow port.

Table 113: Options tab

Option	Description
Allow Merge or upsert	Specifies whether the Optimizer may use a MERGE statement to improve the performance of auto correct load functionality. *Yes*: Allows the Optimizer to consider using a MERGE statement during an auto correct load operation. If the Optimizer does not use a MERGE statement, it uses a stored procedure to identify, insert, and update rows. *No*: The Optimizer will not use a MERGE statement to improve auto correct load performance. The default is Yes.

Table 114: Bulk loader tab

Option	Description
Bulk load	Indicate the bulk load method. Choose one of the following options: *CLI load*: Use the DB2 CLI load utility. The CLI load utility performs better than the load utility because it writes data directly from memory to the DB2 target table or view. You must use DB2 version 8.x or later. *Import*: Use the DB2 import utility to bulk load data. The import utility uses a SQL INSERT statement to write data from an input file into a table or view. *Load*: Use the DB2 bulk load utility. The load utility improves performance over the import utility by writing data directly into the data file. *No*: Do not bulk load data.
Generate files only	This option is available only when *Bulk load* is set to *import* or *load*. Select this check box to generate a data and control file. Rather than loading data into the target shown in the data flow, SAP Data Services generates a control file

Option	Description
	and a data file that you can later load using DB2 bulk loading. This option is useful when the DB2 server is located on a system running a different operating system than the Job Server. The software writes the data and control files in the bulk loader directory specified in the datastore definition. If you have not specified a bulk loader directory, the software writes the files in the **<DS_COMMON_DIR>**`\log\bulkloader` directory. To load the data, you must manually copy the files to the remote system and start the bulk load execution. When you select this check box, only the *Text delimiter* and *Column delimiter* options are available.
Clean up bulk loader directory after load	Select this check box to delete all files in the bulk loader directory after the load completes successfully. If you have not specified a bulk loader directory in the *Connections* tab in the datastore definition, the software writes the files in the **<DS_COMMON_DIR>**`\log\bulkloader` directory. When this option is selected, the software deletes the following files after each bulk load unless an error has occurred: ● Message file (`.log` file name) that DB2 creates for the import, load, or CLI load ● Control file (`.ctl`) that the software generates only when *Bulk load* is set to *import* or *load* ● Data file (`.dat`) that the software generates only when *Bulk load* is set to *import* or *load* ● Bad file (`.bad`) that DB2 generates only when *Bulk load* is set to *load* and the *Data file on client machine* option is not checked. If the *Data file on client machine* option is checked, DB2 creates the `.bad` file on the DB2 server working directory (specified in the *DB2 Properties* tab of the datastore definition). In this case, the software does not delete the .bad file when the bulk load completes.
Mode	Specify the mode for loading data in the target table. Available modes depend on the bulk load method. Available modes when *Bulk load* is set to *import*: *Insert*: Adds new records to the table. Use when loading data into an empty table or when appending data to an existing table that contains data that you want to maintain. *Insert-update*: If a record with matching primary keys exists in the table, updates that record; otherwise, adds new record to the table. This method requires that the target table has primary keys. *Replace*: Deletes all existing records in the table, then adds new records. *Truncate*: Deletes all existing records in the table, then adds new records. Available modes when *Bulk load* is set to *CLI load* or *load*:

Option	Description
	Insert: Appends the new records into the target table. *Replace*: Deletes the existing records, and then inserts the loaded data.
Rows per commit	Enter the number of rows that will be loaded before a commit takes place. If no value is entered, the load utility uses the default value at run time. This option is available only when *Bulk load* is set to *import*.
Save count	Enter the minimum number of rows loaded before the load utility establishes a consistency point. This is the point when DB2 saves the data. DB2 converts this value to a page count, and rounds up to intervals of the extent size. If you enter zero, the load utility establishes no consistency points. This option is available only when *Bulk load* is set to *load*.
Warning row count	Enter the number of warnings allowed for each load operation. This option is available only when *Bulk load* is set to *CLI load* or *load*.
Text delimiter	Enter a single character string delimiter. The default value is a double quote ("). The specified character is used in place of double quotation marks to enclose a character string. The specified character can be any printable or non-printable ASCII character, escaped with a double slash "\\". This option is available only when *Bulk load* is set to *import* or *load*.
Column delimiter	Enter a single-character column delimiter. The default value is a comma (,). This option is available only when *Bulk load* is set to *import* or *load*.
Maximum bind array	Enter the maximum number of rows extracted or transformed before the software sends the data to the DB2 table or view. If you do not enter a value, the software uses the default DB2 CLI Loader value which is 10000. This option is available only when *Bulk load* is set to *CLI load*.
Exception table name	Enter the table into which the DB2 server loads rows that violate a table constraint. Rows that violate constraints are deleted from the target table and inserted into the exception table. This option is available only when *Bulk load* is set to *CLI load* or *load*.
Recoverable	Select this check box to support data recovery through the DB2 roll-forward recovery feature. When this option is not selected, you cannot recover from failure using DB2 roll-forward. When this option is selected, DB2 writes a backup copy of the loaded data. You can use DB2 roll-forward recovery after failure. You must specify the directory for writing the backup file (*Copy target directory*). Select this option only if your DB2 target database is roll-forward enabled. This option is available only when *Bulk load* is set to *CLI load* or *load*.

Option	Description
Copy target directory	Enter the directory where copy files are stored when the *Recoverable* option is enabled. Only local media is supported. This option is available only when *Bulk load* is set to *CLI load* or *load*.
Data file on client machine	Select this check box to have the load utility process the local file directly rather than using FTP to send the data file to the DB2 server. To use this option: ● You must use DB2 version 7.x or later. ● The target DB2 cannot be a DB2 enterprise (extended edition environment). ● The target table and database must not be partitioned. This option is applicable only if the software and DB2 are on different servers. This option is available only when *Bulk load* is set to *load*.

Related Information

Common target table options [page 963]

3.2.2.23.3.3 Hadoop Hive Adapter Target

You can set the following options on the *Adapter Target* tab of the target table editor.

Option	Possible values	Description
Append	True, False	Select *True* to append new data to the table or partition. Select *False* to delete all existing data, then add new data.
Clean up working directory	True, False	Select *True* to delete the working directory after the job completes successfully.
Dynamic partition	True, False	Select *True* for dynamic partitions. Hive evaluates the partitions when scanning the input data. Select *False* for static partitions. Only all-dynamic or all-static partitions are supported.
Number of loaders	Positive integers	Enter a positive integer for the number of loaders (threads). Loading with one loader is known as *single loader loading*. Loading when the

Option	Possible values	Description
		number of loaders is greater than one is known as *parallel loading*. You can specify any number of loaders. The default is 1.

3.2.2.23.3.4 HP Neoview target table options

The following table contains option and description information specific to HP Neoview target tables.

> Note
>
> You can use substitution parameters for all of the options listed below with the exceptions of *Clean up bulk loader directory after load*, *File options*, *Use the control file and Generate files only*, *Mode*, and *Operation*.

Table 115: Bulk Loader Options

Option	Description
Bulk load	Select to use HP Neoview bulk loading options to write the data.

Table 116: Options

Option	Description
Clean up bulk loader directory after load	Deletes all bulk loader-related files (script, data files, temporary file, control file, error file, data files, log file) after the load is complete. If an error occurs during bulk load, SAP Data Services does not delete script and data files. Errors usually occur when: ● There is a syntax error in the script. ● Error tables are not empty. Error tables contain rows that cannot be inserted into the target table due to data conversion or constraint violation.
Column delimiter	Specify a single-character to delimit (enclose) the columns. Default value is /127 (non-printable character). You may also have a comma, semicolon or any decimal character (for example /38 is the decimal value for &), except for the following: any alphabetic characters a-z, A-Z, decimal numbers 0-9, carriage return (/10) or line feed (/13). Note The software uses decimal numbers, and HP Neoview uses octal numbers for delimiter values.
File option	Specify the staging type before loading data into the target: ● Data file ● Named pipe

Option	Description
Generates files only	When this option selected, the software generates a data file and a script file, and ignores the *Number of loaders* option (in the Options tab). You can load the file using HP Neoview bulk loading later. This option is often useful when the target database is located on a system running a different operating system than the Job Server. The software writes the data and control files in the bulk loader directory (default value is <**`<DS_COMMON_DIR>`**\log\bulkloader) specified in the datastore definition. The naming conventions for these files is: **`<DatastoreName_OwnerName_TableName_PID_n>`**`.ctl` and **`<DatastoreName_OwnerName_TableName_PID_n>`**`.dat` where: ● **`<OwnerName>`** is the table owner ● **`<TableName>`** is the target table ● **`<PID>`** is the process ID ● **`<n>`** is a positive integer, optionally used to guarantee that pre-existing files are not overwritten
Mode	Specify the mode for loading data in the target table: ● *Append*: Adds new records to the table ● *Truncate*: Deletes all existing records in the table, then inserts the loaded data as new records
SQL Operation	Specify the type of SQL operation to perform: ● *Insert*: Places data into a Neoview table. ● *Update*: Updates data in a Neoview table. ● *Upsert*: Updates the data in a Neoview table if the data exists, otherwise it inserts the data.
Text delimiter	Specify a character used to delimit (enclose) char or varchar data. Valid values are single quotes, double quotes, or blank (default).

Table 117: Neoview options

Option	Description
Bad data file name	Specify a name and location for source records that fail internal processing before being written to the database. For example, a record that contains 6 fields when 8 fields are expected will fail internal processing and be placed in this file. If this option is left blank during runtime, the software places the file **`<datastore_schema_tablename_PID_n>`**`_badrecord.dat` in the directory listed in **`<DS_COMMON_DIR>`**`\log\bulkloader`.
Enable triggers	Allows triggers (a template SQL statement) to run on a table while loading jobs. When this option is set to true, one of the following situations must also be true: ● update trigger on a table with primary key, enable triggers is before update, and operation must be set to update.

Option	Description
	- update trigger on a table with multiple column primary key, enable triggers is before update, and operation must be set to update.
Failed data file name	Specify a name and location for source records that have a valid format, but could not be written to the database. For example, a record that fails a data conversion step or violates a uniqueness setting is placed in this file. If this option is left blank during runtime, the software places the file `<datastore_schema_tablename_PID_n>_failedrecord.dat` in the directory listed in `<DS_COMMON_DIR>\log\bulkloader`.
Force staging	Specify whether to write rows to an internal staging table before populating the target tables. i Note Using staging tables increases the performance for load operations. However, if the load fails, it may be more difficult to recover.
Maximum number of discarded rows	Specify the maximum number of rows that do not match the expected format or information specified for the table being loaded. When the value is exceeded, the bulk loader stops. Set this option to 0 to ignore rejected rows, continue processing, and log warnings.
Maximum number of failed rows	Specify the maximum number of rows that fail to load into the database. When the value is exceeded, the bulk loader stops. Set this option to 0 to ignore rejected rows, continue processing, and log warnings.
NoAudit	Specify whether to use non-transactional database operations to improve performance. If you set this option to true, you must also set the *sorted by primary key* option and the *truncate* option to true, and the *force staging* option to false.
Number of data connections	Specify the number of data connections to access the database. By default, there is one data connection for every four partitions of the table. i Note The maximum value is limited to the number of partitions for the table.
Retry attempts	Specify the number of times to connect to a database or open a named pipe on behalf of a job.
Retry duration	Specify the amount of time, in seconds, between attempts to connect to the database or to open the named pipe.
Row set size	Specify the number of records in each batch of rows that Neoview Transporter exchanges with the HP Neoview database. If this option is not set, Neoview Transporter selects an optimized value based on the HP Neoview SQL table or query.

Option	Description
Rows per commit	Specify the maximum number of rows to process during a single transaction. When this number is reached, Neoview Transporter will stop processing the job.
Server processes on a connection	Specify the number of Neoview server processes on a single connection. In most instances, set this number to the number of Neoview segments.
Sorted by primary key	Specify whether to sort the source files by the table's primary key. Neoview Transporter can improve performance and parallelism of certain database operations when the data is sorted by the primary key.
Time out	Specify the amount of time, in seconds, for Neoview Transporter to wait for the load operation to complete before timing out and returning an error. The duration starts after a named pipe has successfully opened. The default value is 1000.

Related Information

Common target table options [page 963]

3.2.2.23.3.5 Informix target table options

The following table contains option and description information specific to Informix target tables. All other option information for target tables can be found in the *Common target table options* [page 963].

i Note

Commit at the end of INSERT..SELECT option is not applicable.

Table 118: Options

Option	Description
Drop and re-create table	Drops the existing table and creates a new one with the same name before loading.

Table 119: Bulk Loader Options

Option	Description
Bulk load	Select this check box to use Informix bulk loading options to write the data.
Generate files only	Select this check box to generate a data and control file. Rather than loading data into the target shown in the data flow, SAP Data Services generates a control file and a data file that you can later load using Informix bulk loading. This option is often useful when the Informix server is located on a system running a different operating system than the Job Server.

Option	Description
	The software writes the data and control files in the bulk loader directory specified in the datastore definition. If you have not specified a bulk loader directory, the software writes the files in the `<DS_COMMON_DIR>\log\bulkloader` directory. You need to copy the files to the remote system manually. The files names are `<tablename>.ctl` and `<tablename>.dat`, where `<tablename>` is the name of the target table.
Lock table	Select this check box to lock the table for the duration of the load.
Clean up bulk loader directory after load	Select this check box to delete all bulk load-oriented files after the load is complete, unless an error occurs. When you select this option, the software deletes these files after a successful bulk load: ● Control file ● Log file ● Bad file
Mode	Select the mode for loading data in the target table: ● *Append*: Adds new records to the table. ● *Replace*: Deletes all existing records in the table and then adds new records.
Bulk loader server name	Enter the name of the Informix database server.
Bulk loader database name	Enter the name of the target information warehouse database.
Rows per commit	Enter the number of rows that must be loaded before a commit takes place.
Field delimiter	Enter the character that separates columns. Make sure the character you designate is not used in any of the data columns.
Maximum rejects	Enter the maximum number of acceptable warnings. Bulk load stops after this many warnings. Set this parameter when you expect no warnings, but want to verify that the correct file and table are used. If you enter 0, or do not specify a value, the load continues regardless of the number of warnings issued. The default value is 10.
Begin/end column character	Enter the character that designates the beginning or ending of the column.

Related Information

Common target table options [page 963]

3.2.2.23.3.6 Microsoft SQL Server target table options

The following table contains option and description information specific to Microsoft SQL Server target tables. All other option information for target tables can be found in the *Common target table options* [page 963].

Table 120: Options

Option	Description
Allow merge or upsert (This is available for Microsoft SQL Server 2008 and higher only)	Specifies whether the Optimizer may use a MERGE statement to improve the performance of auto correct load functionality. *Yes*: Allows the Optimizer to consider using a MERGE statement during an auto correct load operation. If the Optimizer does not use a MERGE statement, it uses a T-SQL block to identify, insert, and update rows. *No*: The Optimizer will not use a MERGE statement to improve auto correct load performance. The default is Yes. Note When the data flow contains a Query transform with an *Order by* clause, the Optimizer always uses a T-SQL block to identify, insert, and update rows.

Table 121: Bulk loader options

Option	Description
Bulk load	Select this check box to use Microsoft SQL Server bulk loading options to write the data.
Rows per commit	Select this option to load a specified number of rows per commit.
Mode	Specify the mode for loading data in the target table: ● *Append*: Adds new records to the table. ● *Truncate*: Deletes all existing records in the table, and then adds new records.
Maximum rejects	When the value for *Maximum rejects* is exceeded, the bulk loader stops. Set this parameter when you expect no warnings about rejected rows, but want to verify that the correct file and table are being loaded. If you enter `0` or do not specify a value, the load continues regardless of the number of warnings issued. A rejected row contains data that does not match the expected format or information specified for the table being loaded.
Network packet size	Specify the network packet size in KB. Default is 4 KB. When loading, the client caches rows until it either fills a network packet or reaches the commit size (regardless of whether the packet is full). Then the client sends the packet to the server. You can affect performance by tuning commit size and network packet size.

3.2.2.23.3.7 Netezza target table options

The following table contains option and description information specific to Netezza target tables. All other option information for target tables can be found in the common table options (see related links below).

i Note

All options under the Transaction Control category are not available for Netezza (Include in Transaction, Transaction ID, Commit at the end of INSERT...SELECT).

Table 122: Bulk Loader options tab

Option	Description
Bulk load	Select a bulk-loading method: *File*: Writes all data to a file before loading through the external table to the staging table. For files that are smaller than 4 GB in size, select this option for faster performance. *Named pipe*: Streams data as it is written to the named pipe through the external table to the staging table. For files that are larger than 4 GB in size, select this option for faster performance. *None*: Does not use bulk loading (default).
Mode	Specify the mode for loading data to the target table: *Append*: Adds new records to table (default). *Truncate*: Deletes all existing records in table then adds new records.
Update method	Specify how to apply UPDATE SQL. *Delete-insert*: Issues a DELETE to the target table for data that matches the old data in the staging table, and then issues an INSERT with the new data (default). Select this option for faster performance. i Note Do not use this option if the update rows do not contain data for all columns in the target table because SAP Data Services will replace missing data with NULLs. *Update*: Issues an UPDATE to the target table.
Maximum rejects	Enter the maximum number of error records allowed before the software terminates the job. Default is 10.
Commit size	Enter the maximum number of rows loaded to the staging and target tables before saving the data. A value of 0 means the software loads and saves all rows automatically (default). This option overrides the *Rows per commit* setting.
Text delimiter	Enter the character used to delimit (enclose) char or varchar data. Valid values are single quotes, double quotes, or blank (default). If you expect the data to contain single or double quotes, use an Escape character.
Field delimiter	Enter the character that separates the columns in a row. Valid values are all printable characters except the single quote. The default is pipe (|). If you expect the field delimiter character to be present in the varchar or char data, use a Text delimiter.

Option	Description
Generate files only	Select this check box to generate data into a file on the Job Server. This option is useful when you want to manually load data using the SQL statements generated in the .sql script. Default is cleared.
Escape character	If you expect the Text delimiter character or a row delimiter (the ASCII character line feed, LF) to be present in any varchar or char data, enter a backslash (the only valid value) to escape the text or row delimiter before writing to the file or named pipe. (The software also escapes back slashes in the data with a back slash.) If you expect a NULL string as part of varchar data, then you should set the escape character. The NULL string is used as null indicator to identify null values while bulk loading. Note The null indicator string is not case sensitive so if you have varchar data as NULL (case insensitive) the escape character should be set. Default is cleared because this option can degrade job execution performance significantly.
Clean up bulk loader directory after load	Select this check box to delete all bulk-loader-related files (for example dat, sql, nzlog, nzbad, etc.) after the load completes successfully. If an error occurs during the bulk load, the Netezza server creates an nzbad file in the Database server working directory defined in the datastore editor. If you have enabled FTP by configuring the FTP options in the datastore editor, the software transfers the nzbad and nzlog files from the Database server working directory to the Bulk loader directory on the Job Server computer. Default is checked.

Related Information

Common target table options [page 963]

3.2.2.23.3.8 ODBC target table options

The following table contains option and description information specific to ODBC target tables. All other option information for target tables can be found in the *Common target table options* [page 963].

Table 123: Bulk loader options tab

Note

Bulk loader options are only visible if you are loading to a Netezza target.

Option	Description
Rows per commit	Enter the number of rows that will be loaded before a commit takes place. If no value is entered, the load utility uses the default value at run time. This option is available only when *Bulk load* is set to *import*.

Related Information

Netezza target table options [page 980]

3.2.2.23.3.9 Oracle target table options

The following table contains option and description information specific to Oracle target tables. All other option information for target tables can be found in the *Common target table options* [page 963].

Table 124: Options

Option	Description
Allow merge	Specifies whether the Optimizer may use a MERGE statement to improve the performance of auto correct load functionality. *Yes*: Allows the Optimizer to consider using a MERGE statement during an auto correct load operation. If the Optimizer does not use a MERGE statement, it uses a PL/SQL block to identify, insert, and update rows. *No*: The Optimizer will not use a MERGE statement to improve auto correct load performance. The default is Yes.

Table 125: Bulk loader options tab

Option	Description
Bulk load	Select a bulk loading method. Use database tools to load data in bulk instead of using SQL statements. *API*: Allows you to use an Oracle direct-path load API to load table data directly to database files. The target database must be Oracle 8.1 or later. *File*: Allows you to use a staging file and the Oracle SQL*Loader to load table data. To use this feature, the version of the Oracle SQL*Loader (specified on this tab) and the database (specified in the datastore for the target) must match. If you also want to use the direct-path load method, manually select it from the File Options section. Otherwise, a convention load is performed. Conventional loads generally are slower than direct-path loads because data is loaded to tables rather than directly to database files associated with tables

Option	Description
	None: Allows you to use normal load functionality. See the Options tab.
Mode	Specify the mode for loading data in the target table. Available modes when Bulk load is set to File: *Append*: Adds new records containing the loaded data. *Insert*: Adds new records containing the loaded data. Requires that the table is empty before loading. SQL loader terminates with an error if the table is not empty. *Replace*: Deletes all existing records in the table, and then adds new records containing the loaded data. *Truncate*: Deletes all existing records in the table, and then adds new records containing the loaded data. This mode does not execute any delete triggers. Available modes when Bulk load is set to API: *Append*: Adds new records containing the loaded data. *Truncate*: Deletes all existing records in the table, and then adds new records containing the loaded data.
Rows per commit	Specifies the transaction size in number of rows or bulk loading. If Rows per commit is set to 1000, a commit is sent to the underlying database every 1000 rows. If you do not enter a value, the default (1000) is used. Bulk loading is not available for targets in a real-time jobs.
Maximum rejects	Enter the maximum number of error records allowed before the job is terminated. If you do not enter a value, the default (10) is used.
Recoverable	Select this check box to log direct-path information about the loaded data in the Oracle redo log. Note All conventional loads are automatically specified as recoverable.
Use local index to rebuild primary key	(API method only) Select this option to ensure that the primary key uses the partitioned index (if the table has local partitioned indexes on primary key) instead of any other indexes that are available on primary key.
*SQL *Loader version*	(File method only) The version used to load data into the table. The version of the Oracle SQL*Loader and the database (specified in the datastore for the target) must match.
Text delimiter	(File method only) Enter the character used to delimit char or varchar columns. The default character is a double quotation mark (“). Make sure the character you enter is not used in any of the data columns.

Option	Description
Field delimiter	(File method only) Enter the character used to separate columns. The default character is a comma (,). Make sure the character you enter is not used in any of the data columns. You can specify a non-printable character by entering the ASCII equivalent, such as: **`/ASCII_number`**
Maximum bind array	(File method only) Enter the maximum bind array. The bind array needs to be large enough to contain a single row. For good performance, make this large enough to hold 100 rows. If you do not enter a value, the default Oracle Bulk Loader value is used.
Use the control file	(File method only) Select this check box to load data from a specific bulk loading control file and data file. Rather than loading data from the source shown in the data flow, Data Services directs Oracle to load data from the data file associated with the named control file. Enter the name of the control file. Do not include the `.ctl` extension in the file name. If you do not specify a complete path, Data Services searches for the file in: ● The path you have specified as the bulk loader directory in the datastore definition ● `\$LINK_DIR\log\bulkloader` Both a control file and an associated data file must be in the same directory. If you select this option, you can also select these options or specify these values: ● Direct path ● Clean up bulk loader directory after load ● Rows per commit ● Maximum rejects ● Maximum bind array A variable can also be used.
Generate files only	(File method only) Select this check box to have Data Services generate a data and control file. Rather than loading data into the target shown in the data flow, Data Services generates a control file and a data file that you can later load using Oracle bulk loading. This option is often useful when the target database is located on a system running a different operating system than the Data Services Job Server. Data Services writes the data and control files in the bulk loader directory specified in the datastore definition. You need to copy the files to the remote system manually. Data Services names these files **`<tablename>`** `.ctl` and **`<tablename>`** `.dat`, where **`<tablename>`** is the name of the target table.
Direct path	(File method only) Select this check box to specify a direct-path load.

Option	Description
	To use direct-path load, the version of SQL*Loader available to the Job Server executing the job must be the same as the target database version. For example, you cannot perform a SQL*Loader Version 7.1.2 direct path load to load into a Oracle Version 7.1.3 database. For more information, see the Oracle server documentation.
Clean up bulk loader directory after load	(File method only) Select this check box to have Data Services delete all bulk loader-related files (control file, datafile, log file) after the load is complete. If an error occurs during the bulk load, Data Services creates a `.bad` file and does not delete any files. Errors occur when: ● Log file was not created ● Log file contains “ORA-” or “SQL*Loader-”
Trailing nullcols	(File method only) Select this check box to indicate that columns not represented in the data being loaded should be treated as null columns. Use when a data record is not complete but the existing data needs to be loaded. If this option is not selected, the system generates an error.

3.2.2.23.3.10 SAP HANA target table options

The following table contains option and description information specific to SAP HANA target tables.

Table 126: Options

Option	Description
Table type	For template tables, select the appropriate table type for your SAP HANA target: *Column Store* (default) *Row Store*

Table 127: Bulk loading

Option	Description
Bulk load	Select to enable bulk loading.
Mode	Specify the mode for loading data to the target table: *Append*: Adds new records to the table (default). *Truncate*: Deletes all existing records in the table and then adds new records.
Commit size	The value *default* means Data Services identifies the SAP HANA target table type and applies a default commit size for the maximum number of rows loaded to the staging and target tables before saving the data (committing): *Column Store*: commit size is 10,000 *Row Store*: commit size is 1,000 You can also type a custom value in the field (any value greater than 1).

Option	Description
Update method	Specify how the input rows are applied to the target table: *Default*: Select to let Data Services apply the default value for this option based on the SAP HANA target table type: ● *Column Store* tables use UPDATE. ● *Row Store* tables use DELETE-INSERT. *UPDATE*: Issues an UPDATE to the target table. *DELETE-INSERT*: Issues a DELETE to the target table for data that matches the old data in the staging table, and then issues an INSERT with the new data. Note Do not use DELETE-INSERT if the update rows do not contain data for all columns in the target table, because Data Services will replace missing data with NULLs.

Related Information

Common target table options [page 963]
Performance Optimization Guide: Using Bulk Loading, Bulk loading in SAP HANA [page 2176]

3.2.2.23.3.11 SAP Sybase ASE target table options

The following table contains option and description information specific to SAP Sybase ASE target tables. All other option information for target tables can be found in the common table options (see related links below).

Table 128: Bulk loader options

Option	Description
Mode	Specify the mode for loading data in the target table: *Append*: Adds new records to the table. *Truncate*: Deletes all existing records in the table and then adds new records.
Bulk load	Select this check box to use SAP Sybase ASE bulk loading options to write the data.
Rows per commit	Select this option to load a specified number of rows per commit.
Maximum rejects	When the value for Maximum rejects is exceeded, the bulk loader stops. Set this parameter when you expect no warnings about rejected rows, but want to verify that the correct file and table are being loaded. If you enter `0` or do not specify a value, the load continues regardless of the number of warnings issued.

Option	Description
	A rejected row contains data that does not match the expected format or information specified for the table being loaded.
Network packet size	Specify the network packet size in KB. Default is 4 KB. When loading, the client caches rows until it either fills a network packet or reaches the commit size (regardless of whether the packet is full). Then the client sends the packet to the server. You can affect performance by tuning commit size and network packet size.

Related Information

Common target table options [page 963]

3.2.2.23.3.12 SAP Sybase IQ target table options

The following table contains option and description information specific to SAP Sybase IQ target tables.

Table 129: Bulk loader options

Option	Description
Bulk load	Select this check box to use SAP Sybase IQ bulk loading options to write the data.
SAP Sybase IQ checkpoint	If selected, Data Services enables the SAP Sybase IQ checkpoint as part of the LOAD TABLE SQL statement used to execute the bulk load. If cleared (the default), the checkpoint is not enabled.
Binary format	Select the check box to load the staging and target tables in binary format. This format generally provides faster performance. Clear the check box to load the staging and target tables in delimited format. This format is useful if you want to be able to view the data.
Ignore conversion error	Select to set the SAP Sybase IQ database option CONVERSION_ERROR to OFF. If cleared, Data Services executes with the current database value for the option. For more information, see the SAP Sybase IQ documentation. Conversion errors are reported as warnings in Data Services.
Mode	Specify the mode for loading data into the target table: *Append*: Adds new records to the table. *Truncate*: Deletes all existing records in the table and then adds new records.
Block size (bytes)	The number of bytes read per block. Inappropriate adjustments to this option can dramatically affect performance. Defaults to 500,000 bytes.

Option	Description
File Options	
Field delimiter	This is the delimiter that separates the columns in a row. It can only be a single character (including a non-printable character) and can be represented by a string of ASCII numbers preceded by a forward slash (for instance /95 results in an underscore character while /09 represents a tab). Default value is /127.
Row delimiter	A character sequence that indicates where one row of data ends and the next begins. It can only be a single character; however, you can represent it with a string of up to 4 hexadecimal ASCII characters. Defaults to \n. Field and row delimiters cannot be the same value.
Generate files only	If checked, the software generates the data files in the bulk loader directory specified in the datastore editor and does not execute the bulk load. If cleared (the default), the software generates the data files and executes the bulk load.
Clean up bulk loader directory after load	If checked (the default), the software deletes the data file and auxiliary files after successfully completing the load. If cleared or if the load does not complete successfully, the data file and auxiliary files remain in the bulk loader directory.
Error Handling	
Constraints	The error handling table lets you select which types of constraint violations to ignore and whether to log the errors for each type. Constraint types include *Unique*, *Null*, *Data value*, *Foreign key*, and *All*. For *All*, type the maximum number of constraint violations the software ignores before stopping the job.
Ignore errors	Type the maximum number of violations the software encounters for any given constraint before stopping the job. Typing 0 means tolerate all errors of that constraint (allows unlimited errors).
Log errors	Select the type(s) of constraint violations to log in the message and row log files.

Related Information

Common target table options [page 963]

3.2.2.23.3.13 Teradata target table options

The following table contains option and description information specific to the *Bulk Loader Options* for Teradata target tables. For information on other target table options, see *Common target table options* [page 963].

On the *Bulk Loader Options* tab, the *Bulk loader* choices are:

- FastLoad
- MultiLoad
- TPump
- Parallel Transporter
- Load Utility
- None

The options on the *Bulk Loader Options* tab vary depending on the *Bulk loader* selected. However some options are common to all methods.

FastLoad, MultiLoad, TPump, and Parallel Transporter bulk loaders include several *Attributes*. A different set of attributes displays depending on the bulk loader (and operator) selected. Attribute names in bold indicate that a value is required. You can accept the default values or modify them.

For details on all Teradata options and attributes, consult your Teradata documentation.

Table 130: Common Teradata bulk loader options

Option	Description
File option	Choose the type of file that will contain the data to bulk load: • Data File • Generic Named Pipe • Named Pipes Access Module
Data Services options	
Generate files only	When selected, the software generates a data file and a script file and ignores the *Number of loaders* option (on the *Options* tab). Rather than loading data into the target shown in the data flow, the software generates a control file and a data file that you can later load using Teradata bulk loading. This option is often useful when the target database is located on a system running a different operating system than that of the Job Server. The software writes the data and control files in the bulk loader directory (default value is **`<DS_COMMON_DIR>`**`\log\bulkloader`) specified in the datastore definition. You must copy the files to the remote system manually. The naming conventions for these files is: <**`<DatastoreName_OwnerName_TableName_n>`**.ctl and **`<DatastoreName_OwnerName_TableName_n>`**`.dat` where: • **`<OwnerName>`** is the table owner • **`<TableName>`** is the target table • **`<n>`** is a positive integer, optionally used to guarantee that the software does not overwrite a pre-existing file
Clean up bulk loader directory after load	Select to delete all bulk loader-related files (script, data files, temporary file) after the load is complete. If an error occurs during bulk loading, the software does not delete script and data files. Errors usually occur when: • There is a syntax error in the script. • Error tables are not empty. Error tables contain rows that cannot be inserted into the target table due to data conversion or constraint violation.
Mode	Specifies the mode for loading data into the target table:

Option	Description
	Append: Adds new records to the table. *Replace*: Deletes all existing records in the table, and then inserts the loaded data as new records.
Field delimiter	Specifies a single-character field delimiter. Default value is /127 (non-printable character).
Bulk operation	For MultiLoad and TPump and for the Parallel Transporter bulk loader Update and Stream operators, specify the bulk operation to use: *Insert*: Insert rows. *Upsert* : If the row exists, update it; if not, insert it.
Named pipes access module	For bulk loaders FastLoad, MultiLoad, TPump, and Parallel Transporter with the *File Option Named pipes access module* selected, the following *Named pipe parameters* are available.
Named pipe parameters	You can override the default settings for the following Teradata Access Module parameters: ● Logdirectory ● Loglevel ● Blocksize ● FallbackFilename ● FallbackDirectory ● SignatureChecking The Teradata Access Module creates a log file to record the load status. The Access Module log file differs from the tbuild log that you specify in the *Log directory* option. The Teradata Access Module writes information to fallback data files. If the job fails, the Teradata Access Module uses its log file and fallback data files to restart the load. The bulk loader directory is the default value for both Logdirectory and Fallback-Directory. For more information about these Access Module parameters, see your Teradata tools and utilities documentation.

Table 131: FastLoad bulk loader

FastLoad paramaters	Description
Data encryption	Select to encrypt requests.
Print all requests	Allows printing of every request sent to the Teradata database.
Buffer size	Output buffer size, in kilobyte,s used for Teradata FastLoad messages to the Teradata database. The output buffer size and the size of the rows in the Teradata FastLoad table determine the maximum

FastLoad paramaters	Description
	number of rows that can be included in each message to the database. A larger buffer size reduces processing overhead by including more data in each message. The default buffer size is also the maximum size allowed: 63 kb. If a value greater than 63 kb is specified, Teradata FastLoad: ● Responds with a warning message ● Resets the buffer size back to the default value ● Continues with the Teradata FastLoad job
Character set	Character set specification for the target.

Table 132: MultiLoad bulk loader

MultiLoad parameters	Description
Reduced print output	Select to limit the print output to the minimal information required to determine the success of the job.
Data encryption	Select to encrypt requests.
Character set	Character set specification for the target.

Table 133: TPump bulk loader

TPump parameters	Description
Reduced print output	Select to limit the print output to the minimal information required to determine the success of the job.
Retain macros	Select to keep macro(s) that TPump generates between jobs.
Data encryption	Select to encrypt requests.
Print all requests	Select to allow printing of every request sent to the Teradata database.
Number of buffers	Optionally increase the number of buffers per session from the default of 2 to a maximum of 10.
Periodicity value	This parameter is in effect whenever the Teradata BEGIN LOAD command uses the RATE parameter to control the rate at which statements are sent to the database. The default periodicity value is four 15-second periods per minute. To improve TPump work flow, adjust to a value from 1 to 30.
Character set	Character set specification for the target.
Configuration file	A file that contains various configuration and tuning parameters for TPump.

Table 134: Parallel Transporter bulk loader

Option	Description
Operator	Parallel Transporter operator values include: - Load - Update - Stream - SQL Inserter Note that the *Attributes* vary depending on the *Operator* selected. Refer to your Teradata documentation for information on attributes.
Number of instances	Specify the number of instances for the load operator. This information is included in the Parallel Transporter script that Data Services generates.
Number of DataConnector instances	For the File Option *Data File*, specify the number of DataConnector instances for the read operator to read data files generated by Data Services. This information is included in the Parallel Transporter script that Data Services generates. The value should be less than or equal to the number of data files.
tbuild parameters	
Debug all tasks	Enables debug trace functions for all tasks. Using this option outputs termination return codes that help with script debugging. Corresponds to the tbuild -d option.
Trace all tasks	Enables the trace option for all tasks. If not specified, trace is disabled. Corresponds to the tbuild -t option.
Latency interval (sec)	Latency is the interval value, in seconds, between the flushing of stale buffers. Corresponds to tbuild -l option.
Checkpoint interval (sec)	Specifies a time interval, in seconds, between checkpoints. Defaults to10 seconds. Corresponds to tbuild -z option.

Table 135: Load Utilities

Option	Description
Command line	Use this field to call a custom script. Load utilities include FastLoad, MultiLoad, and TPump. For example for FastLoad, you could enter: `fastload < C:\Teradata\FLScripts\myScript.ctl` Data Services does not parse or modify these scripts.
Named pipe name	For a Load Utility, if you choose either *Named Pipes Access Module* or *Generic Named Pipes* file option, enter the pipe name. On UNIX, the pipe is a FIFO (first in, first out) file that has name of this format: `/temp/`**`<filename>`**`.dat` On Windows, the pipe name has this format: `\\.\pipe\`**`<datastorename_ownername_tablename_loadernum>`**`.dat`

Related Information

Common target table options [page 963]
Teradata [page 902]
Teradata source [page 952]
Performance Optimization Guide: Bulk Loading and Reading, Bulk loading and reading in Teradata [page 2178]

3.2.2.23.4 Target Data_Transfer files and tables

When you add a Data_Transfer transform to a data flow, you create a target for the temporary storage that SAP Data Services uses to process large amounts of data.

3.2.2.23.5 Target XML files, messages, and templates

An XML Schema or DTD format can be added to a job as a target. Choose *Make XML File Target* or *Make XML Message Target* from the context menu that opens when you drop either format into the workspace.

You can also create an XML file target without creating a format by using an XML template.

Option	Description
Make port	Select this check box to make the target file an embedded data flow port.
XML file	(File targets only) The location relative to the Job Server of a file to use as the target. If the file does not exist, SAP Data Services creates it. If the file exists, the software clears the content of the file before writing the output to it. i Note If your Job Server is on a different computer that the Designer, you cannot use Browse to specify the file path. You must type the path. You can type an absolute path or a relative path, but the Job Server must be able to access it. A variable can also be used.
XML test file	(Message targets only) The location relative to the Job Server of a file to use as the message target when you run the job in test mode. If the file does not exist, the software creates it. If the file exists, the software clears the content of the file before writing the output to it. i Note If your Job Server is on a different computer that the Designer, you cannot use Browse to specify the file path. You must type the path. You can type an absolute

Option	Description
	path or a relative path, but the Job Server must be able to access it. A variable can also be used.
Delete and recreate file	Allows you to override the default behavior which is to append new data sets to the file. If selected, the software deletes the old file and creates a new one containing only the current data set.
Print comment	Allows you to include or exclude a comment in the target file data that identifies the data as having been processed by the software.
Replace NULL or blank	Allows you to specify a value that will replace NULL or blank values in element data. Select the check box, then enter a value in the *With* field.
Enable validation	Enable a comparison of the outgoing data to the stored XML Schema or DTD from which this XML file was created. When this option is enabled, the data flow throws an exception if the outgoing data is not valid. When you are developing real-time jobs, this validation helps you ensure sample data is both valid and well-formed. If you turn on this option in production, make sure to include appropriate error handling either in the job or the client application to process an error caused if the real-time job receives data that does not validate against the imported XML Schema or DTD.
Include schema location	Allows you to include or exclude the schema location in the target file data. This check box is selected by default. If you do not want to include the schema location in the XML output, clear this check box.
Include DTD	(For targets created from DTD formats only) The content of an XML target does not normally include the DTD format (which the software uses internally). If you want to add the DTD format to the target file or message, select this check box.
XML encoding	Select an XML encoding for the XML target file. If you do not select a value, the encoding in the XML header field is used. If that field is empty, UTF-8 is used. XML file targets can be saved with a different encoding/code page than the software's system locale. XML message source and target encodings default to UTF-8 and cannot be changed.
XML header	You can use a unique header for each file target. To use this option, you must first enter the header information you want to use for the target. Thereafter, you can edit it from this field. For example, if your header includes more information than the XML Schema version and the encoding, you may want to view and edit this information in the Designer. If you only need to change the XML encoding for this file target, use the XML encoding option instead of editing the header.
DTD file in DOCTYPE	(For targets created from DTD formats only) The content of an XML target does not normally include the DTD format (which the software uses internally). If you want to add a DOCTYPE element to the target file or message, that specifies a path to a DTD format enter the path here or use the Browse button to select one.
Format Name	(Read only) The name of the DTD or XML Schema format used in the Designer.

Option	Description
Root element name	(Read only) The name of the root element used in the DTD or XML Schema.
Namespace	(Read only) The name space used in the XML Schema.

The validation for an XML target allows columns and nested tables marked as optional in the output schema to not be present in the input schema. At run-time the XML target will handle missing columns appropriately.

Related Information

Designer Guide: Embedded data flows [page 392]
XML template [page 1020]
Locales and Multi-byte Functionality [page 1793]

3.2.2.24 Target Writer migrated from Data Quality

If you migrate a Data Quality repository to SAP Data Services and you have projects that contain database Writer transforms, the resulting SAP Data Services jobs will contain migrated Writer targets. A migrated Writer target contains the SQL statements from the Data Quality Writer transform options *Write_SQL_Statement*, *Pre_Write_SQL_Statement*, and *Post_Write_SQL_Statement*.

For more information, see the *Data Services Upgrade Guide*.

3.2.2.25 Template table

Class

Reusable

Access

- To insert as a target, open a data flow diagram in the work space, click the template table icon in the tool palette, and click anywhere in the data flow.
- To insert as a source, open the object library, click the *Datastores* tab, select the desired template table, and drag into the data flow.
- To view options, click the name of the template table in the workspace or in the project area. This opens the object editor.

Description

Template tables are new tables you want to add to a database. You can use a template table one time as a target and multiple times as a source. You cannot use a template table in an ABAP data flow.

A template table provides a quick way to add a new target table to a data flow. When you use a template table, you do not have to specify the table's schema or import the table's metadata. Instead, during job execution, SAP Data Services has the DBMS create the table with the schema defined by the data flow. After you create a template table as a target in one data flow, you can use the same template table as a source in any other data flow.

Use template tables in the design and testing phases of your projects. You can modify the schema of the template table in the data flow where the table is used as a target. Any changes are automatically applied to any other instances of the template table. During the validation process, the software warns you of any errors, such as errors that result from changing the schema.

Before you can use a template table as a source in a data flow design, the data flow where the template table was created as a target has to be valid and you have to save the data flow.

Before executing any job where a template table is used as a source, you must execute the job where the template table is used as a target at least one time. If the template table is used as a target and a source in the same job, then the data flow where it is used as a target must be executed first.

When running a job where a template table is used as a target, use care if the table already exists. If the *Drop and re-create table* option is selected in the template table editor (this is the default option), the software drops the existing table and creates a new one. If the *Drop and re-create table* option is not selected, the software attempts to load data in the existing table. In this case, the software generates run-time errors if the existing table schema does not match the schema generated in the data flow.

When used as a target, the options available from the target editor for template tables are the same as those available for target tables with some exceptions.

Table 136: Target tab

Option	Description
Table name	The name of the table. Can contain alpha, numeric, and underscores; cannot include blank spaces.

Table 137: Options tab

Option	Description
Column comparison	Drops the existing table and creates a new one with the same name before loading.

Option	Description
Drop and re-create table	Drops an existing table with the same name before creating the table specified by the template table. When using template tables in real-time jobs, deselect this and the *Delete data from table before loading* option. These options are selected by default when you create a template table.
Use NVARCHAR for VARCHAR columns in supported databases	Creates nvarchar columns in the template table for all varchar columns in the input schema of the data flow. The data type displays as varchar in the Designer, and, when supported by the DBMS, as nvarchar in the database table. If you are using an ODBC datastore to connect to Oracle, in the datastore editor for *ODBC Miscellaneous* ▸ *NVARCHAR type name*, select *NVARCHAR2*. See also *ODBC* [page 885]. The following database management systems do not support the nvarchar data type: ● DB2 (non-UTF-8) ● Oracle 8.x ● Informix ● SAP Sybase SQL Anywhere ● SAP Sybase ASE ● SAP Sybase IQ For these DBMSs, the software creates columns with varchar data types and increases the column size using a codepage conversion factor based on the client code page defined in the datastore. ⚠ Caution Data loss may occur when transcoding from one national language to a different national language. Data loss will not occur when transcoding from a national language to Unicode. ⚠ Caution Data truncation occurs when the column size of the source exceeds the maximum size allowed by the target DBMS.

Table 138: Bulk Loader Options tab

Option	Description
Bulk load	Not available for template tables.

Table 139: Load Triggers tab

Option	Description
On operation	Not available for template tables.

Before running production jobs, execute the job to load the target table if you have not already done so, then right-click the template table in the object library or in a data flow and select Import Table. The software creates the table in your database and imports it. All information about the table is marked as part of the database and you can make no further changes to the schema. You can now use the new table in expressions, functions, transform options, or for bulk loading. Other features, such as exporting an object, are available for template tables.

Related Information

Target [page 960]

Message processing [page 945]

3.2.2.26 Transform

Class

Reusable

Access

In the object library, click the *Transforms* tab.

Description

Transforms define your data transformation requirements. Transforms use the operation codes associated with each row of data read from a source. The descriptions of individual transforms indicate which operation codes the transforms ignore or use.

Transforms have the following built-in attributes:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.

Attribute	Description
Description	Your description of the transform. Description is not available for query transforms.

If you delete a user-defined transform from the object library, calls to the object are replaced with an icon indicating that the calls are no longer valid, and it is deleted from the project area.

Related Information

Descriptions of transforms [page 1068]

3.2.2.27 Try

Class

Single-use

Access

With a work flow diagram in the workspace, click the try icon in the tool palette.

Description

A try is part of a serial sequence called a try/catch block. Use a single try with each try/catch block; there can be more than one catch with a single try. The try/catch block allows you to specify alternative work flows if errors occur while SAP Data Services is executing a job. Try/catch blocks "catch" classes of errors, apply solutions that you provide, and continue execution.

Do not reference output variables from a try/catch block in any subsequent steps if you plan on using the automatic recovery feature. Referencing such variables could alter the results during automatic recovery.

Tries have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

Related Information

Catch [page 846]

3.2.2.28 While loop

Class

Reusable

Access

With a work flow diagram in the workspace, click the while loop icon in the tool palette.

Description

A while loop repeats a sequence of steps as long as a condition is true.

For each while loop, specify:

- Condition: In the *While* box enter a Boolean expression that the job evaluates.
  The expression must evaluate to TRUE or FALSE. You can use constants, functions, variables, parameters, and standard operators to construct the expression.
- Set of steps: In the while loop workspace, enter the steps you want completed when the condition is true.
  You can add any objects valid in a work flow, including scripts, work flows, and data flows. Connect these objects to represent the order that you want the steps completed.

  > Note
  >
  > Though you can include the parent work flow in the while loop, recursive calls can create an infinite loop.

Related Information

Designer Guide: While loops [page 335]
Smart editor [page 1021]

3.2.2.29 Work flow

Class

Reusable

Access

- In the object library, click the *Work Flows* tab.
- With a job or work flow diagram in the workspace, click the work flow icon in the tool palette.

Description

A work flow contains data flows and the operations that support data flows. The work flow defines the execution order of the data flows and supporting operations. A job is also a work flow.

You can define parameters to pass values into the work flow. You can also define variables for use inside the work flow.

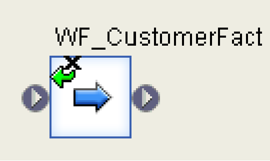
The definition of a work flow can contain the following objects:

- Other work flows
- Data flows
- Scripts
- Try/catch blocks
- Conditionals
- While loops

In some cases, steps in a work flow depend on each other and should always be executed together. You can designate such a work flow (batch jobs only) as a "recovery unit." When designated as a recovery unit, the entire work flow must complete successfully during execution. If any step in such a work flow does not complete successfully, SAP Data Services re-executes all steps in the work flow during automatic recovery, except for ABAP data flows, the software re-executes data flows that executed successfully earlier. The software may or may not re-execute ABAP data flows.

To designate a work flow as a recovery unit, Access work flow Properties, select *Regular* from the *Execution type* dropdown list and select the *Recover as a unit* check box.

On the workspace diagram, a symbol indicates when a work flow is a recovery unit.



3.2.2.29.1 Executing jobs only once

You can ensure that a job executes a work flow only one time by selecting *Regular* from the *Execution type* dropdown list and selecting the *Execute only once* check box on the data flow Properties window. When you select this check box, SAP Data Services executes only the first occurrence of the work flow and skips subsequent occurrences in the job. You might use this feature when developing complex jobs with multiple paths, such as those containing try/catch blocks or conditionals, and you want to ensure that the software executes a particular work flow only once.

Before selecting the *Execute only once* option, note that:

- If you design a job to execute the same *Execute only once* work flow in parallel flows, the software only executes the first occurrence of the work flow, and you cannot control which one the software executes first. Subsequent flows wait until the software processes the first one. The engine provides a wait message for each subsequent work flow. Since only one Execute only once work flow can execute in a single job, the engine skips subsequent work flows and generates a second trace message for each, "Work flow n did not run more than one time. It is an execute only once flow."
- If you design a job to execute more than one instance of the same *Execute only once* work flow, you must manage the values of output variables. The software only processes one such work flow per job. Subsequent instances of the work flow do not run and do not affect the values of variables in the job.
- The *Execute only once* work flow option does not override the *Enable recovery* job option.

Work flows have several built-in attributes.

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the work flow.
Date_created	The date that the object was created.

If you delete a work flow from the object library, calls to the object are replaced with an icon indicating that the calls are no longer valid.

3.2.2.30 XML file

Class

Single-use

Access

In the object library, click the *Formats* tab.

Description

An XML file object allows you to indicate a source or target in a batch or real-time job. When used as a source, an XML file object translates incoming XML-formatted data into an internal SAP Data Services data set. When used as a target, an XML file object translates the data produced by a data flow, including nested data, into an XML-formatted file.

The data read into or written out of an XML file must have a single row at the top-level table. When writing out an empty nested table, the software includes a single row of the nested table, with null values in each column of the table.

To produce the metadata that describes the data that an XML file object handles, the software reads an XML Schema or DTD. The metadata is stored in the repository as either an XML Schema or DTD object.

3.2.2.30.1 Source or target

You can insert an XML file into a data flow by dragging either an XML Schema or DTD format from the *Formats* tab of the object library into the workspace of a data flow. When you drop the format in the workspace, SAP Data Services prompts you to specify that the resulting object is a source or target file or a source or target message.

3.2.2.30.2 Parsing input and producing output

Mapping rules govern how SAP Data Services translates an XML Schema or DTD into its internal schema definition and produces XML formatted data from an internal data set.

Related Information

Designer Guide: Formatting XML documents [page 345]

3.2.2.30.3 Properties

XML file properties are the same as those of its format.

Related Information

DTD [page 906]
XML schema [page 1008]

3.2.2.31 XML message

Class

Single-use

Access

In the object library, click the *Formats* tab.

Description

An XML message object allows you to indicate a real-time source or target in a job.

When used as a source, an XML message object translates incoming XML-formatted messages into an internal SAP Data Services data set. When used as a target, an XML message object translates the data produced by a job, including nested data, into an XML-formatted message and sends the message to the Access Server.

When a real-time job contains an XML message source, it must also contain an XML message target.

The data read into or written out of an XML message must have a single row at the top-level table. When writing out an empty nested table, the software includes a single row of the nested table, with null values in each column of the table.

To produce the metadata that describes the data that an XML message handles, the software reads the format for the XML message. The metadata is stored in the repository as an XML Schema or DTD.

Related Information

DTD [page 906]
XML schema [page 1008]

3.2.2.31.1 Source or target

You can insert an XML message into a real-time job by dragging a XML Schema or DTD format from the *Formats* tab of the object library into the workspace of a data flow. When you drop the format in the workspace, you are prompted to specify that the resulting XML message as a source or target.

3.2.2.31.2 Source and Target editors

You can find information about source and target options elswhere in the *Reference Guide*.

Related Information

XML message source [page 955]
Target XML files, messages, and templates [page 994]

3.2.2.31.3 XML test files

During the design phase of your application, you can execute a real-time job in "test mode." In test mode, the real-time job reads messages from an XML test file specified in the source editor, and writes XML-formatted messages to an XML test file specified in the target editor.

3.2.2.31.4 Parsing input and producing output

Mapping rules govern how SAP Data Services translates an XML Schema or DTD into its internal schema definition and produces XML from an internal data set.

See the Nested Data section of the *Designer Guide* for an introduction to the nested relational data model which the software uses to generate an internal hierarchical schema.

Related Information

DTD [page 906]
XML schema [page 1008]

3.2.2.31.5 Properties

XML file properties are the same as those of its format.

Related Information

DTD [page 906]
XML schema [page 1008]

3.2.2.32 XML schema

Class

Reusable

Access

In the object library, click the *Formats* tab, then double-click the XML Schema category.

Description

SAP Data Services supports W3C XML Schemas Specification 1.0. This XML Schema version is documented on the following web site: *www.w3.org/TR/2001/REC-xmlschema-1-20010502/* .

XML Schemas describe the data structure of an XML file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets.

To use XML Schemas, import XML Schema metadata into the software. During import, the software converts the structure defined in the XML Schema into the the software internal schema based on the nested relationship data model.

Related Information

Rules for importing XML Schemas [page 1014]

3.2.2.32.1 Editor

Open the XML Schema editor by double-clicking an XML Schema name in the object library.

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://my-company.com/namespace"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
  <xs:element name="Order">
   <xs:complexType>
    <xs:sequence>
        <xs:element name="OrderNo" type="xs:string" />
     <xs:element name="CustID" type="xs:string" />
     <xs:element name="ShipTo1" type="xs:string" />
     <xs:element maxOccurs="unbounded" name="LineItems">
       <xs:completType>
       <xs:sequence>
       <xs:element name="Item"    type="xs:string" />
       <xs:element name="ItemQty"   type="xs:string" />
       <xs:element name="ItemPrice" type="xs:string" />
         </xs:sequence>
       </xs:complexType>
     </xs:element>
    </xs:sequence>
   <xs:complexType>
  <xs:element>
</xs:schema>
```

The XML Schema editor displays:

- The nested schema. This example shows:
  - The object name is Simple_Order.
  - Columns at the top level are OrderNo, CustID, ShipTo1, and ShipTo2.
  - LineItems is a nested table.
  - Item, ItemQty, and ItemPrice are columns nested one level.
- The XML Format tab shows:
  - Full path to the XML Schema format file in the *Imported from* text box.
  - Root element in XML Schema format in the *Root element name* text box.
  - Namespace of XML Schema format in the *Namespace* text box.

Related Information

Designer Guide: Formatting XML documents [page 345]

3.2.2.32.2 Properties

XML schema property	Description
Name	The name of the format. This name appears in the object library under the Formats tab and is used for sources and targets (XML files or messages) that reference this format in data flows.
Description	Text that you enter to describe and document the XML Schema.
Imported from	The full path to the format. For example, `D:\data\test.xsd`.

XML schema property	Description
Root element name	The name of the primary node you want to import. SAP Data Services only imports elements of the format that belong to this node or any sub nodes.
Namespace	(Optional) The Namespace URL of the root element.

3.2.2.32.3 Attributes supported for XML schemas

Column attribute	Description
Required	Indicates whether this column always has to be mapped (YES/NO)
Native Type	Original data type of the column. Saved as a string during import.
Default Value	Default value for this column.
Fixed Value	The only value the column can have.
Max Value Inclusive Of Min Value Inclusive Of Max Value Exclusive Of Min Value Exclusive Of	Mapped from the following data type constraining facets: MaxInclusive, MinInclusive, MaxExclusive, and MinExclusive.
Pattern	For a string: the pattern to which its value should match.
Enumeration	Contains a list of all possible values separated by vertical bars. For example: "Red | White | Blue Green | Magenta". A string is cut off at 256 characters.
XML Type	Allows you to track whether the column wasan element or attribute in the original XML Schema.
Length	Number of characters in this column.
Max Length	Maximum number of characters allowed in this column.
Min Length	Minimum number of characters allowed in this column.
Namespace	Column namespace.

Nested table attribute	Description
Any One Column	If choice (for example, "white | black | almond"), then SAP Data Services sets the value of Any One Column to YES. If sequence (for example, "first, last, street, city, state") then the value is set to NO. If both are present in the XML Schema, the value is set to NO.
Maximum Occurrence	Indicates maximum number of rows in the table.

Nested table attribute	Description
	If maximum occurrence equals zero, the software indicates that your XML Schema structure is not valid.
Minimum Occurrence	Indicates minimum number of rows that can be in the table.
Optional Table	Indicates that an instance document might not contain this table, but the software still accepts the document as input.

Related Information

Column attributes for tables [page 958]

3.2.2.32.4 Supported XML Schema components

SAP BusinessObejcts Data Services supports nearly all valid XML Schemas.

The software supports features such as abstract types and blocking in that the software will import and accept these features without error. Except for abstract types, this document does not discuss such features in depth as they do not have a direct impact on the ability of the software to support XML Schemas.

The software imports XML Schema data types as well as element and attribute names and their structure. Once imported, double-click an XML Schema format from the object library to view table and column names and structure. From the XML Format editor, right-click a column name and select edit properties, attributes, and data types.

Related Information

Unsupported XML schemas [page 1016]

3.2.2.32.5 Abstract datatypes

When you build an XML file or message target using an XML schema that contains elements with abstract datatypes you must set the correct value for the "xsi:type" attribute to generate valid XML output. As data flow designer, you must know which of the many derived types is correct for any given element.

i Note

By default, all elements with abstract datatypes have an attribute called xsi:type.

When using XML Schemas with namespaces, you must include the right namespace in the type name. Obtain the right namespace tag by reviewing the namespace tags generated by SAP Data Services (typically ns1, ns2, ...) then using the tag that represents the right namespace in which the type exists.

Example:

Assume you have an element called Publication which has an abstract type called PublicationType. When the software imports this element, it will add an extra column called "xsi:type" as a child of Publication. You must then set the expression for this column to be equal to the expected type of the result (for instance, it could be BookType). To add the correct tag, first execute your job and note the generated tag names. For this example, it is ns1 for a namespace called <http://www.bookworld.com/>. So, for this example, the expression of xsi:type would be "ns1:BookType".

3.2.2.32.6 XML Schema elements

The following XML Schema elements are mapped to attributes when they are imported as metadata.

XML Schema Element	Attribute (nested table or column)
All	All. Elements should occur but they can occur in any order. See Choice.
Choice	Any One Column. If the complex type for an element has been specified as choice then an attribute called Any One Column is created and set to YES. If the complex type has been defined with sequence or several nesting levels containing a mix of choice and sequence then the Any One Column table attribute is created and set to NO. `<Sequence>` , `<choice>` , and `<all>` are handled as follows: ● Sequence becomes "Any One Column = NO". Attributes A, B, and C become columns A, B, and C. ● Choice becomes "Any One Column = YES": Attributes A, B, and C become columns A or B or C. ● All becomes "All": B, C, and A or any combination of the three.
Default	Default Value
Enumeration	Enumeration. The value for this attribute is cut off after 256 characters. As a result, all the enumerated values may not be visible.
Fixed	Fixed Value
Length MinLength MaxLength	Length, Min Length, and Max Length.
MaxInclusive MinInclusive MinExclusive MaxExclusive	Max Value Inclusive Of, Min Value Inclusive Of, Min Value Exclusive Of, Max Value Exclusive Of.
MaxOccurs	Maximum Occurrence (only applies to tables).
MinOccurs	Minimum Occurrence (only applies to tables).
Name	Column name

XML Schema Element	Attribute (nested table or column)
Pattern	Pattern
TotalDigits and FractionDigits	None. Digits are handled as decimal data types. .
Type	Saved as the Native Type attribute (string). The Type element is also translated into a data type (usually `varchar`).
Sequence	See Choice.

Related Information

Unsupported XML schemas [page 1016]

3.2.2.32.7 XML Schema attributes

The following XML Schema **`<attributes>`** are mapped to Data Services column attributes when they are imported as metadata.

XML Schema Attribute	Column Attributes
Default	Default Value
Fixed	Fixed Value The only value the column can have.
Name	Column name
Type	Saved as the Native Type attribute (string). The Type element is also translated into a data type (usually `varchar`).
Use	An XML Schema **`<Use>`** attribute with a value of OPTIONAL becomes the **`<Required>`** attribute with a value of NO. An XML Schema **`<Use>`** attribute with a value of REQUIRED becomes the **`<Required>`** attribute with a value of YES.

Related Information

Unsupported XML schemas [page 1016]

3.2.2.32.8 Included XML Schemas

An XML Schema can be extended by including pointers to other XML Schema files. This is done by using `<import>`, `<include>` and `<redefine>` . These elements are defined at the schema level.

The difference between `<include>` and `<import>` is that for `<include>` the name spaces must be identical in both XML Schemas. `<Redefine>` is similar to `<include>` except the caller can redefine one or more components in the related XML Schema.

When you import an XML Schema, SAP Data Services follows the links to included files to define additional metadata. The included schema information is saved in the repository so that at run time there is no need to access these files again. Inclusions can be files or URLs.

3.2.2.32.9 Groups

XML Schemas allow you to group elements and then refer to the group. A similar concept is available for attributes (called an attribute group). In SAP Data Services any reference to a group will be replaced by the contents of that group.

3.2.2.32.10 Rules for importing XML Schemas

SAP Data Services applies the following rules to convert an XML Schema to the software's internal schema:

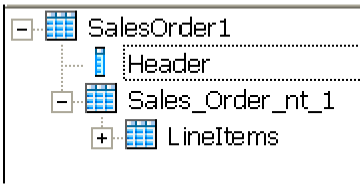
1. Any element that contains an element only and no attributes becomes a column.
2. Any element with attributes or other elements becomes a table.
3. An attribute becomes a column in the table corresponding to the element it supports.
4. Any occurrence of `<choice>` , `<sequence>` or `<all>` uses the ordering given in the XML Schema as the column ordering in the internal data set.
5. Any occurrence of `<maxOccurs ,>` from greater than 1 to “unbounded”, becomes a table with an internally generated name (an implicit table).
   The internally generated name is the name of the parent followed by an underscore, then the string "nt" followed by a sequence number. The sequence number starts at 1 and increments by 1.

After applying these rules, the software uses two additional rules, except where doing so would allow more than one row for a root element:

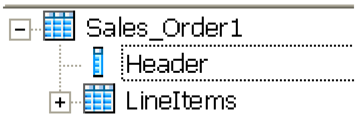
1. If an implicit table contains one and only one nested table, then the implicit table can be eliminated and the nested table can be attached directly to the parent of the implicit table.
   For example, the SalesOrder element might be defined as follows in an XML Schema:

```
<xs:element name="SalesOrder">
<xs:complexType>
<xs:sequence>
<xs:element name="Header"/>
<xs:element name="LineItems" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

When converted in the software, the LineItems element with MaxOccurs ="unbounded" would become an implicit table under the SalesOrder table. The LineItems element itself would be a nested table under the implicit table.



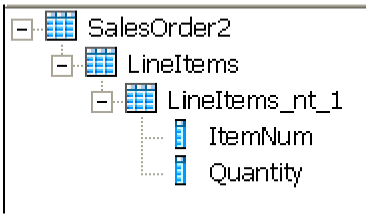
Because the implicit table contains one and only one nested table, the format would remove the implicit table.



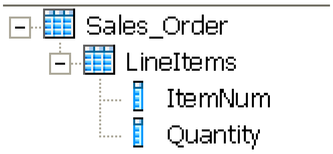
2. If a nested table contains one and only one implicit table, then the implicit table can be eliminated and its columns placed directly under the nested table.
For example, the nested table LineItems might be defined as follows in an XML Schema:

```
<xs:element name="SalesOrder">
<xs:element name="LineItems"  minOccurs="0"
maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element ref="ItemNum"/>
<xs:element ref="Quantity"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

When converted into the software, the grouping with MaxOccurs ="unbounded" would become an implicit table under the LineItems table. The ItemNum and Quantity elements would become columns under the implicit table.



Because the LineItems nested table contained one and only one implicit table, the format would remove the implicit table.



3.2.2.32.11 Unsupported XML schemas

The following XML Schema elements and attributes are not supported in SAP Data Services. They are ignored and not imported.

Component	Description
Annotation	The softwares ignores both documentation and appinfo annotation components.
Non-native attributes	Non-native attributes are attributes that come from a name space other than the one your XML Schema uses. The W3C XML Schema standard enables users to add non-native attributes to attributes and elements. However, the software ignores all such attributes.
XDR files	The software does not support XML Data Reduced (XDR) files. XDR files were used as a format in some products before XML Schema became the standard. There are third-party tools on the market which can automatically convert XDR to XML Schema.

3.2.2.32.12 Limitations

If an XML schema definition contains the following elements or attributes, SAP Data Services imports it with the following limitations:

- Any element or anyAttribute
  You can import an XML schema that contains an Any element or anyAttribute or both, but the format that the software creates does not show the Any element or anyAttribute.
  Consequently, the software ignores the content of the Any element or anyAttribute when it reads an XML instance document. When an element has type anyType, the software treats everything within it as a string and does not recognize the subelements within it.
- Mixed content
  The structure of an XML schema usually consists of elements that contain subelements, and the subelements at the lowest level contain character data. However, an XML schema definition allows character data to appear next to subelements, and the character data is not confined to the lowest level. For instance documents that contain mixed content, the software ignores the character data between any two subcolumns, but captures the values of the subcolumns.

3.2.2.32.13 Data type mappings

SAP Data Services imports data types for XML Schema elements and attributes.

There are two types of built-in data types, the Primitive data types and the Derived data types (derived from primitive). Each data type has the following values defined: space, lexical space, and constraining facet.

If the constraining facet **`<length>`** is missing when metadata is imported into the software, the default varchar(1024) is applied. Similarly, for a decimal the default values 28 and 2 are applied for **`<precision>`** and **`<scale>`** . All other facets like **`<minInclusive>`** , **`<maxInclusive>`** , **`<minLength>`** are imported as column attributes. Enumeration values are also imported as column attributes.

3.2.2.32.14 Primitive types

The table below lists Primitive XML Schema types, examples, and the corresponding data type in SAP Data Services. The constraining facets used are shown in bold.

XML Schema type	Example	Data type
AnyURI	http://www.example.com/	Varchar(len) : len = *length* (in chars)
Base64Binary	GpM7	Varchar(len) : len = *length* (in octets)
Boolean	{true, false, 0, 1}	Varchar(5)
Date	CCYY-MM-DD	Datetime
DateTime	Format = CCYY-MM-DD HH:MM:SS	Datetime
Decimal	7.999	Decimal(p, s) : p = *totalDigits*, a maximum of 28 and s = *fractionDigits*, default =28,2
Double	64 bit floating point	Double (In the software there is no difference between real and double)
Duration	P1Y2M3DT10H30M	Varchar(64)
Float	32 bit floating point, 12.78e-2	Real
gDay		Varchar(12)
gMonth		Varchar(12)
gMonthDay		Varchar(12)
gYear		Varchar(12)
gYearMonth	Gregorian CCYY-MM	Varchar(12)
HexBinary	0FB7	Varchar(len) : len = *length* (in octets)
Notation		N/A
Qname	po:USAddress	Varchar(len) : len = *length* (in chars)
String	"Hello World"	Varchar(len) : len = *length* (in characters)
Time	HH:MM:SS	Datetime

3.2.2.32.15 Derived types

The table below lists pre-defined Derived XML Schema types, examples, and the corresponding data type in SAP Data Services. The constraining facets used are shown in bold.

XML Schema type	Example	Data type
NormalizedString	[No tab/CR/LF in string]	Varchar(len) : len = *length* (in characters)
Token		Varchar(len) : len = *length* (in characters)
Language	En-GB, en-US, fr	Varchar(len): len = *length* (in characters)
NMTOKEN	US, Brésil	Varchar(len): len = *length* (in characters)
NMTOKENS	Brésil Canada Mexique	Varchar(len): len = *length* (in characters)
Name	ShipTo	Varchar(len): len = *length* (in characters)
NCName	USAddress	Varchar(len): len = *length* (in characters)
ID		Varchar(len): len = *length* (in characters)
IDREF		Varchar(len): len = *length* (in characters)
IDREFS		Varchar(len): len = *length* (in characters)
ENTITY		Varchar(len): len = *length* (in characters)
ENTITIES		Varchar(len): len = *length* (in characters)
Integer		Int
NonPositiveInteger		Int
NegativeInteger		Int
Long		Decimal 28,0
Int		Int
Short		Int
Byte		Int
NonNegativeInteger		Int
UnsignedLong		Long
UnsignedShort		Int
UnsigedByte		Int
PositiveInteger		Int
AnyType (ur-type)	unspecified type	Varchar(255)

3.2.2.32.16 User-defined types

User-defined types are XML Schema attributes with a non-XML Schema name space. The XML Schema W3C standard uses a SimpleType element for a user-defined type.

When SAP Data Services finds a user-defined type it finds the base type and uses it to assign a data type for the element. For example: If element X has type TelephoneNumber, its type in the software is `varchar(8)`.

Some simple types are based on other simple types. In such cases the software traces back to the base type.

3.2.2.32.17 List types

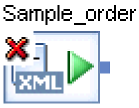
XML Schemas have list types. When it encounters a list, SAP Data Services makes that list's corresponding data type a `varchar(1024)`. All the elements of the list are placed in the value of that column as a string (exactly as it is represented in the XML).

3.2.2.32.18 Union types

A union type enables an attribute or element value to be one or more instances of one type drawn from the union of multiple primitive type and list types. When it encounters a union, SAP Data Services makes that union's corresponding data type a `varchar (1024)`.

3.2.2.32.19 Metadata

If you delete an XML Schema from the object library, XML sources or targets that are based on this format are invalid. SAP Data Services marks the source or target objects with an icon that indicates the calls are no longer valid.



To restore the invalid objects, you must delete the source or target and replace it with a source or target based on an existing XML Schema.

3.2.2.32.20 Error checking

SAP Data Services allows you to control whether it checks each incoming XML file or message for validity against the imported XML Schema. Select *Enable Validation* for an XML source or target in its editor. If you choose to check each XML file or message, the software uses the XML Schema imported and stored in the repository rather than a XML Schema specified by a given XML file or message. If a file or message is invalid relative to the XML Schema, the job produces an error and shuts down.

A typical example of when the software throws validation errors is when either a required element is missing or a new and unexpected element is present in the input. This is true of both the source and target. Consider the following examples:

- A element is defined in the XML Schema with enumeration values of "Black", "White", "StainlessSteal", and "Almond." If the mapping of that element from the XML document yields "Red" that would be incorrect XML. But the XML file target generates the XML regardless. If validation is enabled, then this error is detected.
- If an element's whitespace attribute is set to `collapse`, the software does not change the data in sources or targets to respect this setting. The whitespace attribute is not supported in the software.

During development, you might validate all messages to test for error conditions with representative messages. During production, you might choose to accept rare invalid messages and risk ambiguous or incorrect data.

The software supports XML Schema legal naming such as allowing multiple elements and attributes to have the same name. However, name conflicts should be identified and tested before you import an XML Schema. The software cannot detect naming conflicts and may not report accurate errors which could later lead to runtime errors.

3.2.2.33 XML template

Class

Single-use

Access

To insert as a target:

- Select the XML Template icon in the tool palette, then click the data flow diagram in the workspace.

To view options:

- Click the name of the XML template in the workspace or in the project area. This opens the object editor.

Description

Use an XML template to create an XML file that matches a particular input schema. The XML template does not require and does not produce a corresponding XML Schema or DTD format. Likewise when it generates XML, it does not create column attributes if they are present in its input schema.

Thus, you can use the XML template to produce an XML file without predefining an XML format. You can use an XML template as a target in a batch or real-time job. In an XML template, all data types are converted to varchar.

After adding an XML template to a data flow, specify the name and location of the file. In the XML template target file editor, specify the file in the *XML file* box.

> ℹ Note
>
> When using XML templates in real-time jobs, deselect the *Delete and recreate file* option in the target editor. This option is selected by default when you create an XML target.

Related Information

Target XML files, messages, and templates [page 994]

3.3 Smart editor

This section provides details about options available in the Designer smart editor. Use the smart editor to create scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

Related Information

Scripting Language [page 1709]
Custom functions [page 1694]

3.3.1 Accessing the smart editor

Access and use the embedded smart editor as a pane within any object editor in the Designer, or open the smart editor as a separate, full-size window.

For example, use the *Mapping* tab in a query to access smart editor:

1. Drag a column from an input schema into an output schema to enable the smart editor.

2. Enter text and select options using the smart editor's right-click menu, or click the ellipsis button to open the full-size smart editor window.

   > **i Note**
   >
   > You cannot add comments to a mapping clause in a Query transform. For example, the following syntax is not supported on the Mapping tab:

   ```
   table.column # comment
   ```

   The job will not run and you cannot successfully export it. Use the object description or workspace annotation feature instead.

When you open the smart editor window, the context of the object from which you opened it is displayed in the title bar.

You can open the smart editor from the following locations:

- Query Editor *Mapping* tab
- Query Editor *From* tab
- Query Editor *Where* tab
- Script Editor
- Conditional Editor
- While Loop Editor
- Custom Function Editor
- Function wizard, "Define Input Parameter(s)" page
- SQL - Transform Editor
- Case Editor

3.3.2 Smart editor options

3.3.2.1 Smart editor toolbar

In addition to standard toolbar icons (Open, Save, Print, Undo, Redo, Cut, Copy, Paste, Find, Replace, and Help), the smart editor toolbar also includes special icons to speed up your editing experience. Special smart editor toolbar icons include:

Icon	Description
Show/Hide Editor Library	Toggle to hide or show the Editor Library pane.
Show/Hide keypad	Toggle to hide or show an editing keypad.
Open selection list	Click to show a list of scripting options. Scroll and select the option you want. Double-click or press Enter/Return to add the option to your script.
Function wizard	Click to open the Function wizard window.

Icon	Description
Validate	Click to check your script for errors.

3.3.2.2 Editor Library pane

Use the *Show/Hide Editor Library* icon in the toolbar to show or hide the smart editor library. The library:

- Displays functions, variables, and data using Tabs.
- Allows you to search each tab using the Find option.

3.3.2.2.1 Tabs

The *Functions* tab displays existing functions in SAP Data Services: built-in, custom, and imported.

The *Variables* tab displays variables, parameters, data type formats, and right-click menu options that can be used in the current context. For example, if you open the smart editor to create a custom function, the *Variables* tab will include options on its right-click menu that you can use to insert, delete, and define properties for new parameters and local variables.

The *Data* tab displays the schemas of data flow sources including nested schemas for the current context. For example, if you open the smart editor from an object in a data flow, such as a WHERE clause of a query, schemas are displayed for connected sources. If you open the smart editor from a script object, the *Data* tab is not displayed.

3.3.2.2.2 Find option

You can search or browse through each tab to find the content you want to include in your script or expression.

3.3.2.3 Editor pane

3.3.2.3.1 Syntax coloring

When you type in the editor pane, the text changes color indicating the type of script language element it represents:

- Quoted strings are shown in pink
- Keywords in blue
- Comments in green
- Functions, operators, and variables are shown in black

3.3.2.3.2 Selection list and tool tips

You can use the smart editor with or without showing the editor library. The same list of context-based items available for use when the library is shown is also available in the editor when the selection list is enabled. The selection list shows these items in alphabetical order instead of grouping them into the categories shown in the library. In addition, the selection list displays keywords available for the context in which the editor is opened.

When the selection list is enabled, you can open it from the tool bar. The selection list also opens automatically when it recognizes a string pattern as you type into the editor.

3.3.2.3.2.1 To use the selection list and a tool tip

1. Right-click the editor, view the menu, and make sure that the *Enable Selection List* and *Enable Tool Tip* items are selected.
2. In the editor, enter at least three characters, or the dollar sign variable symbol ($).

   The selection list opens over the editor and highlights the first item (in this alphabetized list) that matches the characters you entered.

   Alternatively, you can click the Open Selection list icon in the tool bar.
3. Double-click an item in the selection list to insert the item in the editor and view an associated tool tip.

   The tool tip displays the same description, definition, or syntax that you would see if the item were selected from the editor library.
4. Complete the script or expression using these tools.

   For example, if you are completing a look-up function, the tool tip will remain on the screen so that you can follow the syntax of the function. If you enter an input value of the wrong data type, the tool tip closes, indicating an error.

3.3.2.3.3 Right-click menu and toolbar

The right-click menu and the tool bar share many commands.

Menu	Toolbar	Key Command
	Open	
	Save As	
	Print	Ctrl + P
Undo	Undo	Ctrl + Z
Redo	Redo	Ctrl + Y
Cut	Cut	Ctrl + X
Copy	Copy	Ctrl + C

Menu	Toolbar	Key Command
Paste	Paste	Ctrl + V
Select All		Ctrl + A
Find	Find	Ctrl + F
Replace	Replace	Ctrl + H
Validate	Validate	
Enable ToolTips		
Enable Selection List		
	Show/Hide Editor library	
	Show/Hide keypad	
	Open selection list	Alt+Down
	Function wizard	
	Help	

Important things to remember:

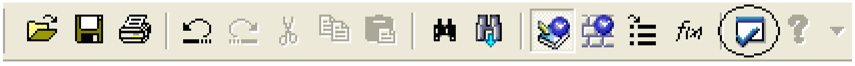
- *Enable ToolTips* and *Enable Selection List* can only be selected using the right-click menu.
- The library, key pad, selection list, and function wizard can be opened from the tool bar.
- Keyboard shortcuts are available for most commands.

3.3.2.3.4 Validation

The smart editor has an embedded error display that includes a script error highlight feature. If validation can occur for the current context, the *Validate* option will be available from the tool bar and right-click menu.

3.3.2.3.4.1 To validate

1. Select the Validate icon in the tool bar or right-click and select *Validate*.



   If errors occur, they are listed in a separate pane below the editor.
2. Double-click each error.
3. The editor redraws to show you where the error occurred in your text.

i Note

If the *Validate* option is not displayed, the expression must be validated in the context of the whole object. Close the full-size smart editor window. The expression is shown on the embedded smart editor. From the Designer menu select ▶ *Debug* ▶ *Validate* ◀.

Related Information

Debugging and Validation [page 1717]

3.3.3 To browse for a function

1. Expand the nodes to find the function you need:
   - Built-in functions are grouped by type
   - Custom functions are listed under the Custom node
   - Imported functions and stored procedures are listed under the name of the datastore used to import them.
2. Click a function and read its description and syntax in the yellow area below the tabs.
3. When you have the function you need, place it into the editor.

3.3.4 To search for a function

1. Select the position in the editor where you want to place the function.
2. In the *Functions* tab of the editor library, select the *Find* node.
3. Enter a string such as `loo`.
4. Press *Enter* or *Tab*.

   All functions that contain the string are returned under the node.
5. To place the function into the editor, do one of the following:
   - double-click
   - drag-and-drop
   - right-click the function and select *Enter*
   - select the function and press *Enter* or *Tab*

3.4 Data Types

Data types are internal storage formats used to store values. A data type implies a default format for displaying and entering values. *Expressions* are a combination of constants, operators, functions, and variables that evaluate to a value of a given data type.

This section discusses how SAP Data Services processes, converts, and evaluates data types.

3.4.1 Descriptions of data types

Data types are internal storage formats used to store values. Data types also imply certain default formats for displaying and entering values. In SAP Data Services:

- Data read from sources is converted to the appropriate SAP Data Services data types.
- Data loaded to targets is converted from their SAP Data Services data types to types appropriate for the target.

The software recognizes the following data types:

- *date* [page 1027]
- *datetime* [page 1029]
- *decimal* [page 1030]
- *double* [page 1030]
- *int (integer)* [page 1031]
- *interval* [page 1031]
- *long* [page 1032]
- *numeric* [page 1035]
- *real* [page 1036]
- *time* [page 1036]
- *timestamp* [page 1037]
- *varchar* [page 1038]

All of these data types allow NULL values.

3.4.1.1 date

The date data type defines calendar dates.

SAP Data Services automatically converts date values to and from the formats used by an external DBMS. Conversion operations from strings to dates, or from dates to strings or numbers require you to specify the format of the date value. To specify a date format, generate a string from the following codes and other literal strings or punctuation.

Date format code	Description	Example
DD	2-digit day of the month value (1-31)	The 2nd day of the month: `02`
MM	2-digit month number (1-12)	The month of March: `03`
MONTH	Full name of the month	The first month of the year: `JANUARY`
MON	Abbreviated, three-character name of month	The first month of the year: `JAN`
YY	2-digit year A `YY` less than 15 is interpreted as being 20 **<yy>** ; for example, 10 would be interpreted as being the year 2010. A `YY` greater than or equal to 15 is interpreted as being 19 **<yy>** ; for example, 35 would be interpreted as being the year 1935. To change the value that the software uses to interpret 2-digit year dates, change the *Century Change Year* value in the Data options. (Select ▶ *Tools* ▶ *Options* ▶ to open the *Options* window, and then select *General* under the *Data* category). The value must be a positive integer between 0 and 99.	The year 1998: `98`
YYYY	4-digit year	The year 1999: `1999`

You can perform various operations on dates such as add and subtract date, datetime, interval, and time values.

The following examples show the use of date formats with functions. The value of the variable `MyDate` is the first day of 1996.

> **Note**
>
> If you use lower case to type “mon” or “month”, the resulting value of `to_char` will be in lower case (For example, jan or january). If you use upper case for “MON” or “MONTH”, the resulting value of `to_char` will be in upper case (For example, JAN or JANUARY).

Example	Output
`to_char($MyDate, 'YYYY.MM.DD')`	1996.01.01

Example	Output
`to_char($MyDate, 'MONTH DD, YYYY')`	JANUARY 01, 1996
`to_char($MyDate, 'DD/MM/YY')`	01/01/96
`to_date('01/01/96','DD/MM/YY')`	1996.01.01 stored as a date
`to_date('01/01/19','DD/MM/YY')`	2019.01.01 stored as a date

Related Information

Date arithmetic [page 1039]

3.4.1.2 datetime

The datetime data type defines calendar dates and times.

SAP Data Services manages date operations in the format used by your DBMS. Conversion operations to or from datetime values require you to specify the format of the datetime. This data type behaves like a concatenation of two data items: The rules for the datetime type are the date rules for the date part, and the time rules for the time part.

If a date field is converted to a datetime value, the default time added to the value is 00:00:00. If a time is converted to a datetime value, the default date added to the value is 0000.01.01. You can also add and subtract date, datetime, interval, and time values.

When converting datetime values to strings, you can choose the sections of the value not to convert by excluding them from the format description. For example, to convert a datetime value to a string containing only the time, specify the function parameters as follows:

```
to_char($MyDateTime, 'hh24:mi:ss.ff')
```

For Oracle, if you load datetime data from SAP Data Services into a char field in an Oracle table, Oracle puts the data in its default datetime format—which includes only date values—and loses the time from the value.

Related Information

Date arithmetic [page 1039]

3.4.1.3 decimal

The decimal data type defines exact decimal numbers.

When specifying a decimal data type in Data Services, you indicate the following characteristics of the type:

Precision: The total number of digits in the value.

Scale: The number of digits to the right of the decimal point.

In the software, the following relations must hold for precision and scale:

1 <= precision <= 96

0 <= scale <= precision

The decimal value can have a plus or minus sign indicating a positive or negative value. The sign can appear before or after the value with any number of blanks between the value and the sign. Unsigned values are considered to be nonnegative. The sign does not count as part of precision.

Leading zeros are permitted in the integer digit, and trailing zeros are permitted in the fraction part.

Input that is more precise than the data type of the column or variable in which it is stored is rounded. Input out of range (absolute value is too large) causes a runtime error.

> i Note
>
> Data Services uses a maximum of 28 precision. Data Services does not enforce precision (that is, having a larger number will not cause an error). Instead, Data Services will round any number more than 28.

The decimal data type and the numeric data type are identical in the software.

> i Note
>
> When you import a table from an Oracle datastore and the native column data type is NUMBER (without any specific precision and scale) the software imports the column as Decimal (28,7) by default. You can override the default Precision and Scale values for an Oracle table at the database level by entering the values in the Advanced section of the Create New DataStore option.

Related Information

Create New Datastore: Oracle options [page 893]

3.4.1.4 double

The double data type defines an 8-byte floating point value, with radix, exponent range, and precision of the platform on which SAP Data Services is running.

3.4.1.5 int (integer)

The int data type defines a 4-byte signed binary integer.

The int value can have a plus or minus sign indicating a positive or negative value. The sign can appear before or after the value with any number of blanks between the value and the sign. Unsigned values are considered to be nonnegative.

3.4.1.6 interval

The `interval` data type defines differences between dates or times. The value is in days unless you specify another unit, such as in a conversion function.

SAP Data Services provides conversion functions to make interval values accessible:`interval_to_char` and `num_to_interval`.

You can add and subtract date, datetime, interval, and time values.

This data type allows NULL values.

Related Information

Date arithmetic [page 1039]

3.4.1.7 Limitations for long and blob

In general, you cannot use long or blob columns in comparisons, calculations, or data type conversions (except for long_to_varchar and varchar_to_long).

Therefore, you cannot use long or blob in the following situations:

- Join, key, compare, or pivot columns
- SQL functions, for example substr
- Expressions and conditions
- SELECT lists of queries containing GROUP BY clauses
- SELECT lists of queries with the Distinct Rows option enabled
- GROUP BY, ORDER BY, or WHERE clauses
- Input or output parameters or return type of functions, for example lookup
- Variable data types
- Work flow and data flow input and output parameters
- Debug filters

The following table shows some of these limitations by transform.

Transform	Do not use long or blob column data in
Case	case condition
Hierarchy_Flattening	parent or child columns
History_Preserving	compare columns
Pivot	pivot transform
Table_Comparison	primary key columns or compare column
Query transform	WHERE, GROUP BY, ORDER BY, or DISTINCT

SAP Information Steward also ignores the auto-correct load option for target tables that contain a long or blob column. The software resets the option and issues a warning message at run time to indicate that the auto-correct load option has been disabled.

i Note

To use large object data types with Informix datastores, you must first configure the Informix ODBC options. For more information, refer to the Informix datastore options.

Related Information

Informix [page 875]

3.4.1.7.1 long

SAP Data Services uses long to represent character-based large objects (clob). The software also converts several other database-specific large object types such as longvarchar and text to the long data type. The Data Services long data type supports mapping from all databases.

The software stores long columns either in memory or in the file system during the data flow execution depending on the size of the long value.

File format considerations for long

You can define long data type columns in Data Services delimited files, XML files, and XML messages. The long data can be in the file or can reference an external file. The notation for this external file is <<**`<filename>`**>>. The software automatically generates the file name.

For example, consider a comma-delimited file format that contains the following database columns:

Column name	Data type
ProductNo	integer

Column name	Data type
Description	long
Picture	blob

In Example 1, the long data appears in the file, but in Example 2, it references a file. The file name designates a path relative to the original input/output file or an absolute path.

Example 1:

```
7369,WidgetA transforms questionable data into trusted sources through a single
environment.,<<pictures\WidgetA.jpg>>
```

Example 2:

```
7499,<<descriptions\WidgetC_descr.txt>>,<<C:\Widgets\pictures\WidgetC.jpg>>
```

Limitations for long

The following limitations exist for long data types:

- The software does not convert between long and any other data types except varchar. You can only convert long to or from varchar using the varchar_to_long() or long_to_varchar() functions.
- Long can be stored in blob.
- When loading a long or longraw column to an Oracle target, the software always extracts and loads the data in separate steps, so it cannot push down the SELECT and load operations in one statement. This restriction does not apply to the Oracle clob, nclob, or blob columns.

Long and blob data types share many of the same limitations (see "Related Topics").

Related Information

Limitations for long and blob [page 1031]
blob [page 1033]
Conversion to or from internal data types [page 1040]

3.4.1.7.2 blob

The binary large object (blob) data type stores any kind of data in binary format. It is commonly used for multimedia data such as images, audio, and video.

Blob columns are stored in the file system during the data flow execution.

File format considerations for blob

You can define blob data type columns in SAP Data Services fixed-width files, delimited files, XML files, and XML messages. You can define an unlimited number of blob columns in a file format, and the blob columns can appear in any order in a file format.

In fixed-width file formats:

- All blob columns are sized in bytes, not characters.
  - The minimum field size of a blob column is 1 byte.
  - The maximum field size is 32,768 bytes.
- Blob data is always inline with the rest of the data in the fixed-width file. The term *inline* means the data itself appears at the location where a specific column is expected.
  For example, consider a fixed-width file format with the following columns:

Column name	Data type
EmployeeNo	integer(4)
LastName	varchar(15)
FirstName	varchar(15)
Password	blob(10)

  In the following sample rows, the blob data (represented by '?'s) appears inline in the file.

7369	Washington	George	??????????
8272	Lincoln	Abraham	??????????

In delimited text files, XML files, and XML messages, blob columns always reference an external file. The notation for this external file is `<filename>`. The software automatically generates the file name.

For example, consider a comma-delimited file format that contains the following three columns:

Column name	Data type
ProductNo	integer
Description	long
Picture	blob

In the following sample row, the blob column references the external file.

```
7369,WidgetA transforms questionable data into trusted sources through a single
environment.,<<pictures\WidgetA.jpg>>
```

You can use blob data types in sources and targets and pass them through transforms.

- When the software reads blob data from fixed-width flat file, it does not trim the leading or trailing 0x00 bytes. It will not treat all 0x00s as Null. They will be stored as is.
- When the software loads a blob into a fixed-width flat file, if the size of the input blob data is not equal to the target blob field size, an error occurs. If the input blob consists of only 0x00s or is a NULL value, the software loads all 0x00s up to the size of the field size of the target blob column.

Database considerations for blob

Data Services can load blob columns to:

- Databases that support parameterized loading.
- Databases with API-based bulk loaders (not file-based bulk loaders), which include:

Database	Bulk loader
Oracle	API
DB2	CLI load
Microsoft SQL Server	Bulk load
SAP Sybase ASE	Bulk load

Limitations for blob

The following limitations exist for blob data types:

- There are no data type conversions between blob and any other data types.
- Blob cannot be stored in long.
- The NULL indicator is not supported for blob data.
- The Data Preview pane in the Designer cannot display blob data.
- The View Data utility shows the data for a blob column as `<blob>`.
- Blob and long data types share many of the same limitations (see "Related Topics").

Related Information

long [page 1032]

Conversion to or from internal data types [page 1040]

Limitations for long and blob [page 1031]

3.4.1.8 numeric

The `decimal` data type and the `numeric` data type are identical in SAP Data Services. See *decimal* [page 1030] for data type information.

3.4.1.9 real

The real data type defines a 4-byte floating point value, with radix, exponent range, and precision of the platform on which SAP Data Services is running.

The real value can have a plus or minus sign indicating a positive or negative value. The sign can appear before or after the value with any number of blanks between the value and the sign. Unsigned values are considered to be nonnegative.

Databases store real values as a 32-bit approximation of the number. Because of this approximation, comparison results are unpredictable when a real value is used in an equality or inequality comparison.

Therefore, it is recommended that you do not use a real value in a WHERE clause. Real values appear in WHERE clauses that the software generates when a column of type real is used:

- As a compare column in the Table_Comparison transform
- In the Map_Operation transform with an opcode of update or delete
- Explicitly in the WHERE clause of a Query transform

In some cases, columns of type real might unintentionally appear in the WHERE clause of these transforms. For example, when no compare columns are specified in a Table_Comparison transform, the transform uses all columns of the table as compare columns. Similarly, if the source of a Map_Operation transform does not have primary key specified and the opcode is update or delete, the transform uses all source columns in the WHERE clause of the UPDATE or DELETE statement.

Use caution when using the `real` data type in these transforms.

For more information, consult the appropriate reference material for Windows NT.

3.4.1.10 time

The time data type defines times of the day, with no calendar date.

SAP Data Services manages time operations in the format used by your database manager. Conversion operations to or from times require you to specify the format of the time value. To specify a time format, generate a string from the following codes and other literal strings or punctuation.

Time format code	Description
HH24	2-digit hour of the day (0-23)
MI	2-digit minute (0-59)
SS	2-digit second (0-59)

The following examples show the use of time formats with functions. The value of the variable `MyTime` is 25 minutes after 8 in the evening.

Example	Output
`to_char($MyTime, 'HH24:MI:SS.FF')`	`20:25:00`

Example	Output
`to_char($MyTime, 'HH24:MI')`	`20:25`

You can add and subtract date, datetime, interval, and time values.

Related Information

Date arithmetic [page 1039]

3.4.1.11 timestamp

The `timestamp` data type supports the `timestamp` (with no zone information) data type in Oracle 9i. The `timestamp` data type incorporates up to a 9-digit sub-second.

Arithmetic

Add or subtract timestamp values. The resulting data type from addition or subtraction operations depends on the operation and data types involved. See *Date arithmetic* [page 1039] for details.

Conversion between timestamp and character strings

You can convert between timestamp values and character values using the to_date and to_char functions. These functions have a format called FF which indicates the sub-second digits. For example, valid function calls are:

```
to_date ('2002.02.26 01234004 09:03:25','yyy.mm.dd ff hh24:mi:ss')

to_char (timestamp_column, 'yyyy.mm.dd hh24:mi:ss.ff')
```

Use the FF format for datetime columns to access sub-seconds. For example, a DB2 timestamp column is mapped to datetime in SAP Data Services. This column contains micro-second. You can access these sub-seconds using the FF format.

Limitations

You cannot use timestamp columns in the SQL transform or in an Oracle stored procedure.

To use a timestamp column in the SQL transform, convert the timestamp column in the select list of the SQL transform to a character format using the to_char function and convert it back to timestamp using the to_date function.

To use a timestamp column in an Oracle stored procedure, convert input and output timestamp parameters in the stored procedure to char, using the to_char function and convert the output parameter back to timestamp in SAP BusienssObjects Data Services using the to_date function. Alternatively, you can convert the input parameter back to timestamp in the stored procedure using the Oracle to_timestamp function.

3.4.1.12 varchar

When specifying a varchar data type, indicate the following characteristic of the type:

Characteristic	Description
Length	Number of characters that the variable or column can hold. Length must be greater than zero. There is no maximum allowable value for the length.

Character strings longer than the number of characters defined for the column or variable are truncated on the right to the length of the data type. Only the required number of characters is used to store strings shorter than length.

SAP Data Services provides functions to convert values to and from strings; to join strings together, use the concatenation operator (||). This data type allows NULL values.

The software conforms to the ANSI SQL-92 varchar standard and treats varchar data as follows:

- Keeps trailing blanks in character values that you insert into varchar columns.
- Keeps trailing blanks when you read from sources with string data types. If you want to remove trailing blanks from your input data, you must use the rtrim or rtrim_blanks function.
- Ignores trailing blanks when you compare varchar data in transforms (Case, Query, and Table_Comparison) and functions (decode, ifthenelse, lookup, lookup_ext, lookup_seq).

  Note

  Not all database servers follow the ANSI standard for trailing blanks in insert, select, and compare operations. Therefore, if the software pushes down the insert, select, and compare operations to the database servers, the operations might return different results than when the software evaluates them. For the most current information on the treatment of trailing blanks, refer to the documentation for the specific database server.

- The ANSI standard treats an empty string as a zero length varchar value.

  Note

  The software treats an empty string differently, depending on the source type. For example, Oracle treats an empty string as a NULL value, but DB2 and Microsoft SQL Server treat an empty string as a zero-length varchar value. For the most current information on the treatment of empty strings, refer to the documentation for your specific database server.

- When using the Equal (=) or Not Equal (<>) operators to compare a value with a NULL constant, the comparison always evaluates to FALSE. Use the IS NULL and IS NOT NULL operators to test for NULL values in the WHERE clause of the Query transform, the lookup_ext function, and the SAP Data Services scripting language.

If you currently run the software pre-version 11.5.0 scripts and data flows, it is recommended that you migrate them to use the ANSI varchar behavior because the previous varchar behavior will not be supported in a future version.

The software supports reading, transforming, and loading National Language Supported (NLS) data from different language locales using the varchar data type. The software supports national character-set data types in the following databases:

Database	Version	National character-set data type
DB2	7.0 and higher	graphic, vargraphic
MS SQL Server	7.0 and higher	nchar, nvarchar
Oracle	9i and higher	nchar, nvarchar2

When the software encounters a national character-set data type in an expression, it binds the column with the data type recommended by the database. Likewise, when using the Metadata Exchange command in the Designer, the data type used in the database (not Data Services' `varchar` data type) is passed on to an SAP BusinessObjects universe.

The engine reads and loads national character-set data types seamlessly without the need for you to configure a locale for a database client and its datastore for the columns that use these data types.

Related Information

NULL values and empty strings [page 1715]

Processing with and without UTF-16 Unicode [page 1799]

3.4.2 Data type conversion

This section discusses how Data Services processes various data types—conversions during arithmetic operations and between data types.

3.4.2.1 Date arithmetic

Data Services performs some implicit data type conversions on `date`, `time`, `datetime`, `timestamp`, and `interval` values when performing date arithmetic. The following table describes these conversions:

Operation	Return type
DATE + INTERVAL	DATE
TIME + INTERVAL	TIME

Operation	Return type
DATETIME + INTERVAL	DATETIME
INTERVAL + INTERVAL	INTERVAL
DATE - DATE	INTERVAL
DATE - INTERVAL	DATE
TIME - TIME	INTERVAL
TIME - INTERVAL	TIME
INTERVAL - INTERVAL	INTERVAL
DATETIME - DATE	INTERVAL
DATETIME - TIME	INTERVAL
DATETIME - DATETIME	INTERVAL
DATETIME - INTERVAL	DATETIME
TIMESTAMP + TIMESTAMP	INTERVAL
TIMESTAMP - TIMESTAMP	INTERVAL
TIMESTAMP + INTERVAL	TIMESTAMP
TIMESTAMP - INTERVAL	TIMESTAMP

3.4.2.2 Conversion to or from internal data types

Data Services performs data type conversions when it imports metadata from external sources or targets into the repository and when it loads data into an external table or file. The software uses its own conversion functions rather than those specific to the database or application that is the source of the data.

Additionally, if you use a template table or Data_Transfer table as a target, the software converts from internal data types to those of the respective DBMS.

3.4.2.2.1 Unsupported data types

Data Services can read, load, and invoke stored procedures involving unknown data types provided that your database servers can convert from VARCHAR to the native (unknown) data type and from the native (unknown) data type to VARCHAR. Data Services might have a problem loading VARCHAR to a physical CLOB column if the native database does not support that conversion (for example, bulk loading or auto-correct load could fail).

When the software encounters a column assigned to an unsupported data type, it does not import the metadata for the column and indicates an error. The file errorlog.txt contains an entry indicating the column that is ignored. To include the column in your job, convert the data type to one supported by the software before importing the metadata for the table.

> ℹ Note
>
> Use the varchar_to_long function to convert a VARCHAR data type to a LONG datatype before loading physical CLOB. If from a prior installation, you are using a VARCHAR column in the physical schema for loading, this will still work.

3.4.2.2.2 Attunity Streams

The following table shows the conversion from Attunity Streams data types to Data Services data types when Data Services imports metadata from an Attunity Streams source or target into the repository.

Attunity Streams data type	Converts to Data Services data type
bigint	double ℹ Note Because int is only four bytes, data is lost during the conversion.
double	double
float	double
long varchar	Converted to long only if imported using an ODBC datastore.
real	real
date	datetime
decimal	decimal
integer	int
smallint	int
time	time
timestamp	datetime
varchar	varchar

3.4.2.2.3 Cobol copybook

The following table shows the conversion from COBOL copybook data types to Data Services data types when Data Services imports metadata from a COBOL copybook source or target into the repository.

COBOL copybook data type	Converts to Data Services data type
computational	decimal
comp	decimal

COBOL copybook data type	Converts to Data Services data type
computational-1	float
comp-1	float
computational-2	double
comp-2	double
computational-3	decimal
comp-3	decimal
computational-4	decimal
comp-4	decimal
computational-5	decimal
comp-5	decimal
computational-6	decimal
comp-6	decimal
computational-x	decimal
comp-x	decimal
binary	decimal
packed-decimal	decimal
float	float
double	double
signed-short	int
unsigned-short	int
signed-int	int
unsigned-int	int
signed-long	int
unsigned-long	int
integer	int
DISPLAY, PICTURE contains A or X (character data)	varchar
Floating point like +9.9(5)E+99	double

3.4.2.2.4 Hadoop Hive

The following table shows the conversion between Hadoop Hive data types and Data Services data types when Data Services imports metadata from a Hadoop Hive source or target.

Hadoop Hive data type	Converts to Data Services data type
tinyint	int

Hadoop Hive data type	Converts to Data Services data type
smallint	int
int	int
bigint	decimal(20,0)
float	real
double	double
string	varchar
boolean	varchar(5)
complex	not supported

3.4.2.2.5 HP Neoview

The following table shows the conversion between HP Neoview data types and Data Services data types when Data Services imports metadata from an HP Neoview source or target into the repository and when it loads data into an external table or file.

HP Neoview data type	Converts to or from Data Services data type
char	varchar
nchar	varchar
national char	varchar
pic x	varchar
varchar	varchar
long varchar	varchar
char varying	varchar
nchar varying	varchar
national char varying	varchar
numeric	decimal
decimal	decimal
smallint	int
integer	int
largeint	double
pic s comp	numeric
pic s	decimal
float	double
double precision	double
real	real

HP Neoview data type	Converts to or from Data Services data type
date	date
time	time
timestamp	timestamp
interval	not supported

The following table shows the conversion from internal data types to HP Neoview data types in template tables or Data_Transfer transform tables.

Data Services data type	HP Neoview data type in template table
blob	not supported
date	date
datetime	timestamp
decimal	decimal
double	double precision
int	int
interval	float
long	varchar
numeric	Decimal: if precision is < = 18 Numeric: if precision is > 18
real	float
time	time
varchar	varchar
timestamp	timestamp

3.4.2.2.6 IBM DB2

The following table shows the conversion between DB2 data types and Data Services data types when Data Services imports metadata from a DB2 source or target into the repository and when it loads data into an external table or file.

DB2 data type	Converts to or from Data Services data type
bigint	int Note Because int is only four bytes, data is lost during the conversion to Data Services data type.
blob	blob

DB2 data type	Converts to or from Data Services data type
character	varchar
clob	long
date	date
dbclob	long
decimal	decimal
double	double
float	double
graphic	varchar
integer	int
long varchar	long
long vargraphic	long
real	real
smallint	int
time	time
timestamp	datetime
varchar	varchar
vargraphic	varchar

The following table shows the conversion from internal data types to DB2 data types in template tables or Data_Transfer transform tables.

Data Services data type	DB2 data type in template table
blob	blob
date	date
datetime	date
decimal	decimal
double	double
int	int
interval	real
long	clob
numeric	character
real	real
time	time
varchar	varchar
timestamp	date

3.4.2.2.7 Informix

The following table shows the conversion between Informix data types and Data Services data types when Data Services imports metadata from an Informix source or target into the repository and when it loads data into an external table or file.

Informix data type	Converts to or from Data Services data type
blob	blob
byte	blob
char	varchar
character	varchar
character varying	not supported
clob	long
date	date
datetime	datetime
dec	decimal
decimal	decimal
double	double
float	double
int	integer
integer	integer
money	decimal
numeric	decimal
real	real
serial	not supported
smallfloat	double
smallint	integer
text	long
varchar	varchar

The following table shows the conversion from internal data types to Informix data types in template tables or Data_Transfer transform tables.

Data Services data type	Informix data type in template table
blob	blob
date	date
datetime	date
decimal	decimal
double	float

Data Services data type	Informix data type in template table
int	int
interval	int
long	clob
numeric	decimal
real	real
time	date
varchar	varchar
timestamp	date

3.4.2.2.8 Microsoft Excel

Microsoft ActiveX Data Objects (ADO) makes it possible to format and convert Excel data sources. The following table shows the conversion from ADO data types to Data Services data types when Data Services imports metadata from an Excel source or target into the repository.

ADO data type	Converted to Data Services data type
adDouble	double
adCurrency	double
adBoolean	varchar
adDate	timestamp
adDBTimestamp	timestamp
ad...Char	varchar

3.4.2.2.9 Microsoft SQL Server

The following table shows the conversion between Microsoft SQL Server data types and Data Services data types when Data Services imports metadata from a Microsoft SQL Server source or target into the repository and when it loads data into an external table or file.

Microsoft SQL Server data type	Converts to or from Data Services data type
binary	not supported
bigint	decimal
bit	int
char	varchar
date (SQL Server 2008 and higher only)	date

Microsoft SQL Server data type	Converts to or from Data Services data type
datetime	datetime
datetime2 (SQL Server 2008 and higher only)	datetime
decimal	decimal
float	double
image	blob
int	int
money/smallmoney	decimal
nchar	varchar
ntext	long
numeric	decimal
nvarchar	varchar
nvarchar(max)	long
real	real
smalldatetime	datetime
smallint	int
text	long
time (SQL Server 2008 and higher only)	time ℹ Note On UNIX, the sub-seconds (HH:MI:SS.`<ff>`, where ff represents the sub-seconds) for the time data type are always set to zero (0). To get a more precise time, use the datetime or datetime2 data type.
timestamp	not supported
tinyint	int
varbinary	not supported
varbinary(max)	blob
varchar	varchar
varchar(max)	long
xml	long

The following table shows the conversion from internal data types to Microsoft SQL Server data types in template tables or Data_Transfer transform tables.

Data Services data type	MS SQL Server data type in template table
blob	image
date	datetime

Data Services data type	MS SQL Server data type in template table
	date (SQL Server 2008 and higher only)
datetime	datetime datetime2 (SQL Server 2008 and higher only)
decimal	decimal
double	float
int	int
interval	real
long	text
numeric	decimal
real	real
time	datetime time (SQL Server 2008 and higher only)
varchar	varchar
timestamp	datetime datetime2 (SQL Server 2008 and higher only)

3.4.2.2.10 MySQL

The following table shows the conversion between MySQL data types and Data Services data types when Data Services imports metadata from a MySQL source or target into the repository and when it loads data into an external table or file.

MySQL data type	Converts to or from Data Services data type
bigint	decimal
decimal	decimal
dec	decimal
bit	int
tinyint	int
bool	int
smallint	int
mediumint	int
int	int
integer	int
year	int

MySQL data type	Converts to or from Data Services data type
float	real
double	double
datetime	datetime
timestamp	datetime
date	date
time	time
varchar	varchar
nvarchar	varchar
nchar	varchar
char	varchar
enum	varchar
set	varchar
tinytext	long
text	long
mediumtext	long
longtext	long
tinyblob	blob
blob	blob
mediumblob	blob
longblob	blob

The following table shows the conversion from internal data types to MySQL data types in template tables or Data_Transfer transform tables.

Data Services data type	MySQL data type in template table
blob	blob
date	date
datetime	timestamp
decimal	decimal
double	double
int	int
interval	float
long	text
numeric	decimal
real	float
time	time

Data Services data type	MySQL data type in template table
varchar	varchar
timestamp	timestamp

3.4.2.2.11 Netezza

The following table shows the conversion between Netezza data types and Data Services data types when Data Services imports metadata from a Netezza source or target into the repository and when it loads data into an external table or file.

Netezza data type	Converts to or from Data Services data type
bigint	decimal
boolean	int
byteint	int
char	varchar
date	date
double precision	double
float	double
integer	int
interval	varchar
nchar	varchar
nvarchar	varchar
numeric	decimal
real	real
smallint	int
time	time
time with time zone	varchar
timestamp	datetime
varchar	varchar

The following table shows the conversion from internal data types to Netezza data types in template tables or Data_Transfer transform tables.

Data Services data type	Netezza data type in template table
date	date
datetime	timestamp
decimal	numeric
double	double precision

Data Services data type	Netezza data type in template table
int	integer
interval	real
numeric	numeric
real	real
time	time
timestamp	timestamp
varchar	varchar

3.4.2.2.12 ODBC

The following table shows the conversion between ODBC data types and Data Services data types when Data Services imports metadata from an ODBC source or target into the repository and when it loads data into an external table or file.

ODBC data type	Converts to or from Data Services data type
bigint	decimal
char	varchar
datalink	not supported
date	date
decimal	decimal
double	double
float	double
graphic	not supported
int	int
nclob	not supported
numeric	decimal
real	real
sql_longvarchar	long
sql_longvarbinary	blob
sql_wlongvarchar	long
time	time
timestamp	datetime
tinyint	int
user-defined	not supported
varchar	varchar

The following table shows the conversion frominternal data types to ODBC data types in template tables or Data_Transfer transform tables.

Data Services data type	ODBC data type in template table
blob	sql_long varbinary
date	date
datetime	timestamp
decimal	decimal
double	double
int	int
interval	real
long	sql_long varchar
numeric	decimal
real	real
time	time
varchar	varchar
timestamp	timestamp

3.4.2.2.13 Oracle

The following table shows the conversion between Oracle data types and Data Services data types when Data Services imports metadata from an Oracle source or target into the repository and when it loads data into an external table or file.

Oracle data type	Converts to or from SAP Data Services data type
char	varchar
blob	blob
clob	long
date	datetime
decimal	decimal
doubleprecision	double
float	double
label	not supported
long	long
long raw	blob
nchar	varchar
nclob	long
number	int: If scale is 0 and precision is < 9

Oracle data type	Converts to or from SAP Data Services data type
	decimal: All other
nvarchar2	varchar
real	double
row	not supported
rowid	not supported
timestamp	timestamp
varchar	varchar
varchar2	varchar

The following table shows the conversion from internal data types to Oracle data types in template tables or Data_Transfer transform tables.

Data Services data type	Oracle data type in template table
blob	blob
date	date
datetime	date
decimal	decimal
double	double
int	int
interval	real
long	clob
numeric	number
real	double
time	date
varchar	varchar2
timestamp	timestamp

3.4.2.2.14 SAP HANA

The following table shows the conversion between SAP HANA data types and Data Services data types when Data Services imports metadata from an SAP HANA source or target into the repository and when it loads data into an external table or file.

SAP HANA data type	Converts to Data Services data type
integer	int
tinyint	int
smallint	int

SAP HANA data type	Converts to Data Services data type
bigint	decimal
char	varchar
nchar	varchar
varchar	varchar
nvarchar	varchar
decimal or numeric	decimal
float	double
real	real
double	double
date	date
time	time
timestamp	datetime
clob	long
nclob	long
blob	blob
binary	blob
varbinary	blob

The following table shows the conversion from internal data types to SAP HANA data types in template tables.

Data Services data type	Converts to SAP HANA data type
blob	blob
date	date
datetime	timestamp
decimal	decimal
double	double
int	integer
interval	real
long	clob/nclob
real	decimal
time	time
timestamp	timestamp
varchar	varchar/nvarchar

3.4.2.2.15 SQL Anywhere

The following table shows the conversion between SAP Sybase SQL Anywhere data types and Data Services data types when Data Services imports metadata from a SQL Anywhere source or target into the repository and when it loads data into an external table or file.

SQL Anywhere data type	Converts to or from Data Services data type
BIT	int
[UNSIGNED] TINYINT	int
[UNSIGNED] SMALLINT	int
INT or INTEGER	int
UNSIGNED INT or INTEGER	decimal(10,0)
BIGINT	decimal(19,0)
UNSIGNED BIGINT	decimal(20,0)
DEC or DECIMAL	decimal
NUMERIC	decimal
FLOAT(1-24) or FLOAT	real
FLOAT(25-53)	double
REAL	real
DOUBLE [PRECISION]	double
SMALLMONEY	decimal(10,4)
MONEY	decimal(19,4)
CHAR	varchar
NCHAR	varchar
VARCHAR	varchar
NVARCHAR	varchar
LONG VARCHAR	long
LONG NVARCHAR	long
TEXT (LONG VARCHAR)	long
NTEXT (LONG NVARCHAR)	long
XML	long
UNIQUEIDENTIFIERSTR (CHAR(26))	varchar(36)
DATE	date
TIME	time
SMALLDATETIME	datetime
DATETIME	datetime
TIMESTAMP	datetime

SQL Anywhere data type	Converts to or from Data Services data type
BINARY	blob
IMAGE	blob
LONG BINARY	blob
VARBINARY	blob
UNIQUEIDENTIFIER	blob

The following table shows the conversion from internal data types to SQL Anywhere data types in template tables or Data_Transfer transform tables.

Data Services data type	SQL Anywhere data type in template table
blob	LONG BINARY
date	DATE
datetime	DATETIME
decimal	DECIMAL
double	DOUBLE
int	INTEGER
interval	REAL
long	LONG VARCHAR/LONG NVARCHAR
real	REAL
time	TIME
timestamp	DATETIME
varchar	VARCHAR/NVARCHAR

Note

When using SQL Anywhere:

- TIMESTAMP supports six-digit sub-seconds (i.e. milliseconds)
- DECIMAL supports precision up to 96 due to ACTADECIMAL_MAXPRECISION (SQL Anywhere supports precision up to 127)
- VARCHAR columns support max length of 32767 characters
- LONG columns support a maximum length of 2GB characters

3.4.2.2.16 SAP Sybase ASE

The following table shows the conversion between SAP SAP Sybase ASE data types and Data Services data types when Data Services imports metadata from a Sybase ASE source or target into the repository and when it loads data into an external table or file.

SAP Sybase ASE data type	Converts to or from Data Services data type
binary	not supported
bit	int
char	varchar
datetime	datetime
decimal	decimal
double	double
float	double
image	blob
int	int
money	decimal(20,4)
numeric	decimal
real	real
smalldatetime	datetime
smallint	int
smallmoney	decimal(12,4)
text	long
timestamp	not supported
tinyint	int
varbinary	not supported
varchar	varchar

The following table shows the conversion from internal data types to SAP Sybase ASE data types in template tables or Data_Transfer transform tables.

Data Services data type	SAP Sybase ASE data type in template table
blob	image
date	datetime
datetime	datetime
decimal	decimal
double	float
int	int
interval	real
long	text
numeric	decimal
real	real
time	datetime

Data Services data type	SAP Sybase ASE data type in template table
varchar	varchar
timestamp	datetime

3.4.2.2.17 SAP Sybase IQ

The following table shows the conversion between SAP SAP Sybase IQ data types and Data Services data types when Data Services imports metadata from a Sybase IQ source or target into the repository and when it loads data into an external table or file.

SAP Sybase IQ data type	Converts to or from Data Services data type
bigint	decimal
binary	not supported
bit	int
blob	blob
char	varchar
clob	long
date	date
datetime	datetime
decimal	decimal
double	double
float	real
int	int
long binary	blob
long varchar	long
money	decimal(19,4)
numeric	decimal
real	real
rowid	decimal
smalldatetime	datetime
smallint	int
smallmoney	decimal(10,4)
time	time
timestamp	datetime
tinyint	int
unsigned bigint	double

SAP Sybase IQ data type	Converts to or from Data Services data type
unsigned int	int
varbinary	not supported
varchar	varchar

The following table shows the conversion from internal data types to SAP Sybase IQ data types in template tables or Data_Transfer transform tables.

Data Services data type	SAP Sybase IQ data type in template table
blob	longbinary
date	date
datetime	timestamp
decimal	decimal
double	double
int	int
interval	float
long	N/A
numeric	N/A
real	real
time	time
varchar	varchar
timestamp	timestamp

3.4.2.2.18 Teradata

The following table shows the conversion between Teradata data types and Data Services data types when Data Services imports metadata from a Teradata source or target into the repository and when it loads data into an external table or file.

Teradata data types	Converts to or from Data Services data type
blob	blob
byteint	int
clob	long
char varying (n)	varchar(n)
char [(n)]	varchar(n)
date	date
decimal	decimal
double precision	float

Teradata data types	Converts to or from Data Services data type
float	float
int	int
long varchar	long
long vargraphic	long
numeric	decimal
real	float
smallint	int
time	time
timestamp	datetime
varchar < 32000	varchar
varchar >= 32000	long
varbyte < 32000	not supported
varbyte >= 32000	blob

The following table shows the conversion from internal data types to Teradata data types in template tables or Data_Transfer transform tables.

Data Services data type	Teradata data type in template table
blob	blob
date	date
datetime	timestamp
decimal	decimal
double	N/A
int	int
interval	N/A
long	long varchar
numeric	N/A
real	N/A
time	time
varchar	varchar
timestamp	N/A

3.4.2.3 Conversion of data types within expressions

When possible, SAP Data Services optimizes data flows by pushing expressions down to an underlying database manager. In a single transaction, the software can push down expressions so that they are performed by the underlying database manager. However, when the software evaluates an expression which includes operands of more than one data type, the software attempts to convert the operands to the same data type first. (Except for national character-set data types which can be pushed down while others in an expression are not. For more information about supported national character-set data types, see *varchar* [page 1038]).

When a conversion is required, the software provides a message at validation.

If the conversion is illegal, the software provides an error and you must remove the mismatch before executing the job.

If the conversion is legal, the software provides a warning indicating that it will not interrupt job execution.

Note

When the software converts a data type to evaluate an expression, the results might not be what you expect. To avoid legal but incorrect conversions, always validate before executing and examine the circumstance of any data type conversion warnings.

3.4.2.4 Conversion among number data types

SAP Data Services uses a type-promotion algorithm to evaluate expressions that contain more than one number data type. Number data types are ranked from highest to lowest, as follows:

- decimal
- double
- real
- int

If the software encounters expressions that have more than one number data type among the operands, it converts all of the operands to the data type of the operand with the highest ranking type.

For example, if A is an `int` and B is a `double`, the expression `A+B` is evaluated by first converting A to `double` and then adding the two `double` values. The result is type `double`.

If in addition to A and B, you multiply the result by a `decimal` number C, then `(A+B)*C` is evaluated by first converting `(A+B)` to decimal, and then performing the indicated operations on the two `decimal` values. The result is type `decimal`.

For addition, subtraction, and multiplication, the operation result will be equal to the higher of the two operands. For example:

```
int + double = double
```

The following algorithm is used for division:

Numerator data type	Denominator data type			
	int	real	double	decimal(p,s1)
int	double	double	double	decimal(p,s1)
real	double	double	double	decimal(p,s1)

Numerator data type	Denominator data type			
	int	real	double	decimal(p,s1)
double	double	double	double	decimal(p,s1)
decimal(p,s2)	decimal(p,s2)	decimal(p,s2)	decimal(p,s2)	decimal(p,max(10,s1,s2))

Conversion among decimals of different scale or precision

If decimals of two different scales are included in a single expression, the software uses the higher of the two scales. For example:

```
decimal(5,4) * decimal(7,2) = decimal(7,2)
```

Expect a loss of precision when operating on two `decimals` of different scale values. For example, when adding a `decimal(28, 26)` to a `decimal(28,1)`, the resulting decimal value has the lower of the two scale values:

```
400000.5 + 40.00005 = 400040.5
```

The least scale for division involving a decimal is 10.

Conversions between strings and numbers

When the software encounters a string where a number would normally be expected (for example, in mathematical operations or functions that expect numeric arguments), it will attempt to convert the string to a number.

For multiplication and division operations, operands are converted to numbers. Other promotion algorithms are shown in the following table.

Table 140: Addition

Provided data type	Data type required to evaluate the expression		
	Number	Date/time	String
Number	OK (promoted)	Number to interval	String to number
Date/time	Number to interval	Illegal	String to interval
String	String to number	String to interval	String to real

Table 141: Subtraction

Provided data type	Data type required to evaluate the expression			
	Number	Date/time	String	Interval
Number	OK (promoted)	Illegal	String to number	Interval to number
Date/time	Number to interval	OK	String to interval	OK

Provided data type	Data type required to evaluate the expression			
	Number	Date/time	String	Interval
String	String to number	String to datetime	String to real	String to number
Interval	Number to interval	Illegal	String to interval	OK

Table 142: Comparison

Provided data type	Data type required to evaluate the expression			
	Number	Date/time	String	Interval
Number	OK (promoted)	Illegal	String to number	OK
Date/time	Illegal	OK	Illegal	Illegal
String	Illegal	String to datetime	OK	String to interval
Interval	OK	Illegal	String to interval	OK

Conversions between strings and dates

For Oracle, if you load datetime data from SAP Data Services into a char field in an Oracle table, Oracle puts the data in its default datetime format—which includes only date values—and loses the time from the value.

3.4.2.5 Conversion between explicit data types

You can use functions to convert data from one type to another:

- *interval_to_char* [page 1578]
- *julian_to_date* [page 1591]
- *num_to_interval* [page 1635]
- *to_char* [page 1676]
- *to_date* [page 1678]
- *to_decimal* [page 1679]

You can also import database-specific functions to perform data type conversions.

3.4.2.6 Conversion between native data types

A Data Quality transform can get and set field data in a format other than the declared data type. For example, if the input field is varchar, it can be mapped to an int data type field, as long as the varchar field contains all digits. However, certain conversions are not supported depending on the data type and field content.

Note

When a data type is mapped to an input or output field that is an invalid data type, the transform issues a verification error.

Example

The USA Regulatory Address Cleanse transform has a varchar type input field named Postcode_Full. Varchar field types can write to any kind of data type as long as the data is formatted correctly, and contains all digits. The Postcode_Full field could be integer because the field contains numbers. However, the Postcode_Full field could not be date type because it does not conform to the date format.

The remaining portion of this section lists each field type and the applicable and invalid data types for each.

3.4.2.6.1 date

Input: A transform can read from a date input field to the following data types:

- date
- character
- datetime

Output: A transform may write to the following data types from a date data type:

- date
- character
- datetime

Invalid input and output data types for date:

- integer
- double
- decimal

3.4.2.6.2 datetime

Input: A transform can read from a datetime input field to the following data types:

- datetime
- character
- date (with possible truncation)
- time (with possible truncation)

Output: A transform may write to the following data types from a datetime data type:

- datetime
- character

- date (with possible truncation)
- time (with possible truncation)

Invalid input and output data types for datetime:

- integer
- double
- decimal

3.4.2.6.3 decimal

Input: A transform can read from a decimal input field to the following data types:

- decimal
- character
- integer (data may be truncated)
- double (data may be truncated)

Output: A transform may write to the following data types from a decimal data type:

- decimal
- character
- integer (with possible truncation)
- double (with possible truncation)

Invalid input and output data types for decimal:

- date
- time
- datetime

3.4.2.6.4 double

Input: A transform can read from a double input field and write to the following data types:

- double
- character
- decimal (with possible truncation)
- integer (with possible truncation)

Output: A transform may write to the following data types from a double data type:

- double
- character
- decimal (with possible truncation)
- integer (with possible truncation)

Invalid input and output data types for double:

- date
- time
- datetime

3.4.2.6.5 int (integer)

Input: A transform can read from an int (integer) input field to the following data types:

- integer
- character
- decimal
- double

Output: A transform may write to the following data types from an int data type:

- integer
- character
- double
- decimal

Invalid input and output data types for int:

- date
- time
- datetime

3.4.2.6.6 varchar

Input: A transform may read input varchar data to any other supported data type. However, if the varchar data is not formatted correctly for the data type, the results are undefined. For example, if a varchar data type is converted to integer, it must contain all digits to convert correctly.

Output: A transform may write any supported data type to a varchar data type. The transform automatically converts the field contents to varchar data.

> Note
>
> The data may be truncated if the output field is not long enough.

3.5 Transforms

A transform is a step in a data flow that acts on a data set. Built-in Data Services transforms are available through the object library.

3.5.1 Operation codes

Data Services maintains operation codes that describe the status of each row in each data set described by the inputs to and outputs from objects in data flows. The operation codes indicate how each row in the data set would be applied to a target table if the data set were loaded into a target. The operation codes are as follows:

Operation Code	Description
NORMAL	Creates a new row in the target. All rows in a data set are flagged as NORMAL when they are extracted by a source table or file. If a row is flagged as NORMAL when loaded into a target table or file, it is inserted as a new row in the target. Most transforms operate only on rows flagged as NORMAL.
INSERT	Creates a new row in the target. Rows can be flagged as INSERT by the Table_Comparison transform to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The Map_Operation transform can also produce rows flagged as INSERT. Only History_Preserving and Key_Generation transforms can accept data sets with rows flagged as INSERT as input.
DELETE	Is ignored by the target. Rows flagged as DELETE are not loaded. Rows can be flagged as DELETE in the Map_Operation and Table Comparison transforms. Only the History_Preserving, transform with the *Preserve delete row(s) as update row(s)* option selected, can accept data sets with rows flagged as DELETE.
UPDATE	Overwrites an existing row in the target table. Rows can be flagged as UPDATE by the Table_Comparison transform to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The Map_Operation transform can also produce rows flagged as UPDATE. Only History_Preserving and Key_Generation transforms can accept data sets with rows flagged as UPDATE as input.

3.5.2 Descriptions of transforms

The transforms described in this section are available from the object library on the *Transforms* tab.

The transforms that you can use depend on the Data Services package that you have purchased. If a transform belongs to a package that you have not purchased, it is unavailable and cannot be used in a Data Services job.

Transforms are grouped into the following categories:

- Data Integrator: Transforms that allow you to extract, transform, and load data. These transform help ensure data integrity and maximize developer productivity for loading and updating data warehouse environment.

- Data Quality: Transforms that help you improve the quality of your data. These transforms can parse, standardize, correct, enrich, match and consolidate your customer and operational information assets.
- Platform: Transforms that are needed for general data movement operations. These transforms allow you to generate, map and merge rows from two or more sources, create SQL query operations (expressions, lookups, joins, and filters), perform conditional splitting, and so on.
- Text Data Processing: Transforms that help you extract specific information from your text. These transforms can parse large volumes of text, allowing you to identify and extract entities and facts, such as customers, products, locations, and financial information relevant to your organization.

Table 143: Data Integrator transforms

Transform	Description
Data_Transfer	Allows a data flow to split its processing into two sub data flows and push down resource-consuming operations to the database server.
Date_Generation	Generates a column filled with date values based on the start and end dates and increment that you provide.
Effective_Date	Generates an additional "effective to" column based on the primary key's "effective date."
Hierarchy_Flattening	Flattens hierarchical data into relational tables so that it can participate in a star schema. Hierarchy flattening can be both vertical and horizontal.
History_Preserving	Converts rows flagged as UPDATE to UPDATE plus INSERT, so that the original values are preserved in the target. You specify in which column to look for updated data.
Key_Generation	Generates new keys for source data, starting from a value based on existing keys in the table you specify.
Map_CDC_Operation	Sorts input data, maps output data, and resolves before- and after-images for UPDATE rows. While commonly used to support Oracle changed-data capture, this transform supports any data stream if its input requirements are met.
Pivot (Columns to Rows)	Rotates the values in specified columns to rows. (Also see Reverse Pivot.)
Reverse Pivot (Rows to Columns)	Rotates the values in specified rows to columns.
Table_Comparison	Compares two data sets and produces the difference between them as a data set with rows flagged as INSERT and UPDATE.
XML_Pipeline	Processes large XML inputs in small batches.

Table 144: Data Quality transforms

Transform	Description
Address Lookup	Completes and populates addresses with minimal data, and can offer suggestions for possible matches.
Associate	Compares group numbers to find associated matches from different Match transforms.
Country_ID	Parses input data and then identifies the country of destination for each record.

Transform	Description
Data_Cleanse	Identifies and parses name, title, and firm data, phone numbers, Social Security numbers, dates, and e-mail addresses. It can assign gender, add prenames, generate Match standards, and convert input sources to a standard format. It can also parse and manipulate various forms of international data, as well as operational and product data.
DSF2 Walk Sequencer	Adds delivery sequence information to your data, which you can use with presorting software to qualify for walk-sequence discounts.
Geocoder	Identifies and appends geographic information to address data such as latitude and longitude.
Global_Address_Cleanse	Identifies, parses, validates, and corrects global address data, such as primary number, primary name, primary type, directional, secondary identifier, and secondary number.
Match	Compares records, based on your criteria, or business rules, to find matching records in your data.
USA_Regulatory_Address_Cleanse	Identifies, parses, validates, and corrects USA address data according to the U.S. Coding Accuracy Support System (CASS).
User_Defined	Does just about anything that you can write Python code to do. You can use the User-Defined transform to create new records and data sets, or populate a field with a specific value, just to name a few possibilities.

Table 145: Platform transforms

Transform	Description
Case	Simplifies branch logic in data flows by consolidating case or decision making logic in one transform. Paths are defined in an expression table.
Map_Operation	Modifies data based on current operation codes and mapping expressions. The operation codes can then be converted between data manipulation operations.
Merge	Unifies rows from two or more sources into a single target.
Query	Retrieves a data set that satisfies conditions that you specify. A Query transform is similar to a SQL SELECT statement.
Row_Generation	Generates a column filled with integer values starting at zero and incrementing by one to the end value you specify.
SQL	Performs the indicated SQL query operation.
Validation	Ensures that the data at any stage in the data flow meets your criteria. You can filter out or replace data that fails your criteria.

Table 146: Text Data Processing transforms

Transform	Description
Entity_Extraction	Extracts information (entities and facts) from unstructured data and creates structured data that can be used by various business intelligence tools.

i Note

For all transforms, to refresh a target schema after making changes to transform options, choose *View* ▸ *Refresh* or press *F5*.

3.5.3 Data Integrator transforms

3.5.3.1 Data_Transfer

Writes the data from a source or the output from another transform into a transfer object and subsequently reads data from the transfer object. The transfer type can be a relational database table or file.

Use the Data_Transfer transform to push down operations to the database server when the transfer type is a database table. You can push down resource-consuming operations such as joins, GROUP BY, and sorts.

Related Information

Performance Optimization Guide: Splitting a data flow into sub data flows [page 2153]
Target options [page 1072]

3.5.3.1.1 Data inputs

The data input is from a source or the output data set from another transform with rows flagged with the NORMAL operation code. This data is referred to as the *input data set*.

The input data set must not contain hierarchical (nested) data.

3.5.3.1.2 Editor

Use the Data_Transfer editor to specify the transfer type and options associated with that type. Depending on the transfer type you select, additional tabs appear.

- When *Transfer type* is `Table` and *Database type* is any RDBMS (such as `Oracle` or `Microsoft SQL Server`), additional tabs are Options, Bulk Loader Options, Pre-Load Commands, and Post-Load Commands.

The Options tab displays the DDL to create the table. You can modify this CREATE TABLE statement to add clauses such as EXTENTSIZE. You can also save this DDL to execute later.

3.5.3.1.3 Target options

The tabs that appears on the target editor depends on the transfer type that you specify.

3.5.3.1.3.1 General tab

Transfer type	Option	Description
File, Table, or Automatic	*Enable transfer*	Enables or disables the execution of the Data_Transfer transform. It is selected by default. You might want to disable this transform if you are tuning performance and you want to see the effect of the Data_Transfer transform. Note When you run the job in debug mode, Data Services automatically disables all Data_Transfer transforms.
File	*File options: File name*	Name of the flat file that you want to use as transfer for sub data flows. The file does not need to exist.
File	*File options: Root directory*	The name of the root directory that will contain the file to use for transfer. If your default Job Server and Designer reside on the same computer, you can use the browse button (ellipses) to find the *Root directory*. If your default Job Server does not reside on your local computer, you must manually enter the path to your *Root directory*. You can use a global variable or parameter for the pathname.
File, Table, or Automatic	*Join rank*	Indicates the rank of the output data set relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. For new jobs, specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.

Transfer type	Option	Description
Table - relational	*Table options: Array fetch size*	Indicates the number of rows retrieved in a single request to a source database. The default value is 1000. Higher numbers reduce requests, lowering network traffic, and possibly improve performance. The maximum value is 5000. This option is available for source tables from DB2, Informix, ODBC, Oracle, and SQL Server datastores. When retrieving a column with an Oracle long data type, Data Services automatically sets Array Fetch Size to 1. If a column has an Oracle long data type, Data Services can only retrieve one row at a time.
Table - relational	*Table options: Database type*	Select an item in the *Database type* box to set the content of additional tabs on the Data_Transfer transform editor to match the specific options for that database type. This option allows you to quickly set target option values in data flows. If your target datastore has multiple configurations, the target editor lists database types that you defined for your datastore configuration. To add or remove items in this list, edit the datastore configuration information in the datastore editor. Data Services allows you to use target table editor option values from any datastore configuration: ● If the datastore has only one configuration, then the initial values for the target table editor are defaults set by Designer for that database type or version. ● If the datastore has more than one configuration and there are different database types/versions, then Data Services determines the initial values for the additional database types/versions from the *Use values from* box in the Create New Configuration dialog (a sub-dialog of the datastore editor). ● If you also select the *Restore values if they already exist* check box (in the Create New Configuration dialog), Data Services looks for previously defined values that once existed for that database type or version. It is possible for a data flow to contain target table editor values for a database type or version, even if its datastore configuration was deleted. Data Services retains all target table editor values saved with every datastore configuration. If such values exist, then it restores those values. Otherwise, it gets the values from the configuration you select from the *Use values from* option.

Transfer type	Option	Description
		For example, suppose you set a configuration for Oracle 8i. When you edit the target table editor options, you change the *Rows Per Commit* default value of 1000 to 500. Later you add a new datastore configuration for a Microsoft SQL Server 2000 database to your original datastore and set the *Use values from* option to Oracle 8i. In this case, the target table editor settings for SQL Server inherit the value 500 for *Rows per Commit* because this was the value set in the Oracle 8i configuration.
Table - relational	*Table options: Table name*	Name of the database table that you want to use as transfer for sub data flows. Specify the table name with the following format: `<datastorename.ownername.tablename>` You can click the browse button (ellipses) to display your datastores. Select a table name from the list or type in the name of a new table.
File, Table, or Automatic	*Transfer type*	Choose one of the following transfer types to temporarily store the data of each sub data flow: ● *Table:* Database table from an existing datastore. If you choose this type, specify the Table options below (*Table name*, *Database type*, and *Array fetch size*). ● *File:* A flat file. If you choose this type, specify the File options below (*Root directory* and *File name*). ● *Automatic:* The Data Services optimizer chooses the transfer type from: ○ Your datastores that selected the *Enable automatic data transfer* check box, or ○ The pageable cache directory that you specify in the Server Manager. The Data Services optimizer chooses the transfer type and location that could provide the optimal performance, based on subsequent operations that the data flow contains. For example, if an ORDER BY follows the Data_Transfer transform, the optimizer might pick the database datastore that contains the data so that the ORDER BY can be pushed down to the database. If the data flow does not contain an ORDER BY, GROUP BY, DISTINCT, join, or any expression that can be pushed down, the Optimizer chooses the pageable cache directory. If multiple files are available (one on

Transfer type	Option	Description
		each job server in a server group), the optimizer chooses the directory that is local to the data flow process.

3.5.3.1.3.2 Options tab

Transfer type	Option	Description
Table - relational database	*Data definition language (DDL)*	You can edit or save the SQL CREATE TABLE statement that Data Services generates. You might want to add extra parameters (such as table space name or extent size) or type in your own DDL statement. Data Services saves the DDL and uses it at job execution time.
Table - relational database	*Delete data before loading*	Deletes the existing data in the table before loading. Defaults to selected. Clear this checkbox to append data to the existing data in the table.
Table - relational database	*Drop and re-create before loading*	Drops the existing table and creates a new one with the same name before loading. Defaults to selected. Note Unlike a template table, you can use bulk loading options for a transfer table even when the *Drop and re-create before loading* is checked.
Table - relational database	*Enable partitioning*	(Displayed only if the transfer table is either physically partitioned or logically partitioned) Enables Data Services to use the partition information in this transfer table. If you select *Enable partitioning*, Data Services transfers data using the number of partitions in the table as the maximum number of parallel instances. If you select *Enable partitioning*, you cannot select *Number of Loaders*.
Table - relational database	*Generate default DDL*	Click this button to display the SQL CREATE TABLE statement that Data Services generates.
Table - relational database	*Number of loaders*	Loading with one loader is known as "single loader loading." Loading when the number of loaders is greater than one is known as "parallel loading." The default number of loaders is 1. You can specify any number of loaders.

Transfer type	Option	Description
		If you select *Number of Loaders*, you cannot select *Enable partitioning*. When parallel loading, each loader receives the number of rows indicated in the Rows per commit option, in turn, and applies the rows in parallel with the other loaders. For example, if you choose a Rows per commit of 1000 and set the number of loaders to 3, the first 1000 rows are sent to the first loader. The second 1000 rows are sent to the second loader, the third 1000 rows to the third loader, and the next 1000 rows back to the first loader.
Table - relational database	*Rows per commit*	Specifies the transaction size in number of rows. If set to 1000, Data Services sends a commit to the underlying database every 1000 rows. This option is not available for targets in real time jobs.

3.5.3.1.3.3 Bulk Loader Options tab

Transfer type	Option	Description
Table - relational database	-	Available options depend on the database in which the table is defined. See *Target tables* [page 963].

3.5.3.1.3.4 Pre-Load Commands and Post-Load Commands tabs

Transfer type	Option	Description
Table - relational database	-	Specify SQL commands that Data Services executes before starting a load or after finishing a load into a transfer table. When a data flow is called, Data Services opens all the objects (queries, transforms, sources, and targets) in the data flow. Next, Data Services runs the target's preload script. Therefore, Data Services executes any preload SQL commands before processing any transform. ℹ Note Because Data Services executes the SQL commands as a unit of transaction, you should not include transaction commands in preload or postload SQL statements.

Transfer type	Option	Description
		Both the *Pre Load Commands* tab and the *Post Load Commands* tab contain a *SQL Commands* box and a *Value* box. The *SQL Commands* box contains command lines. When you first open the tab, an empty line appears. To edit a line, select the line in the *SQL Commands* box. The text for the SQL command appears in the *Value* box. Edit the text in that box. To add a new line, determine the desired position for the new line, select the existing line immediately before or after the desired position, right-click, and choose *Insert Before* to insert a new line before the selected line, or choose *Insert After* to insert a new line after the selected line. Finally, type the SQL command in the *Value* box. To delete a line, select the line in the *SQL Commands* box, right click, and choose *Delete*. You can include variables and parameters in preload or postload SQL statements. Put the variables and parameters in either brackets, braces, or quotes. Data Services translates each statement differently, writing a statement that depends on the variable or parameter type. <table><tr><th>Entered statement</th><th>Variable value</th><th>Written statement</th></tr><tr><td>[$X]</td><td>5</td><td>5</td></tr><tr><td>[$X]</td><td>John Smith</td><td>John Smith</td></tr><tr><td>{$X}</td><td>5</td><td>5</td></tr><tr><td>{$X}</td><td>John Smith</td><td>John Smith</td></tr><tr><td>'$X'</td><td>5</td><td>5</td></tr><tr><td>'$X'</td><td>John Smith</td><td>John Smith</td></tr></table> You cannot use Pre Load and Post Load SQL commands in a real-time job.

3.5.3.1.4 Data outputs

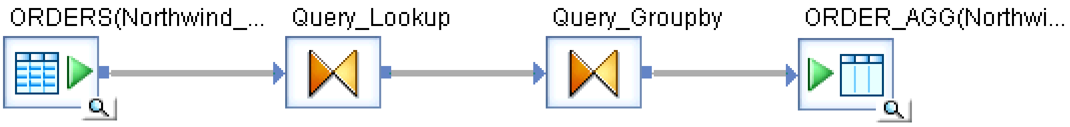
A data set with the same schema and the same operation code as the input data set. If a subsequent ORDER BY or GROUP BY is pushed down to the database, the output rows are in the ORDER BY (or GROUP BY) order.

Data Services automatically splits the data flow into sub data flows and executes them serially. The sub data flow names use the following format, where `<n>` is the number of the sub data flow:

```
<dataflowname_n>
```

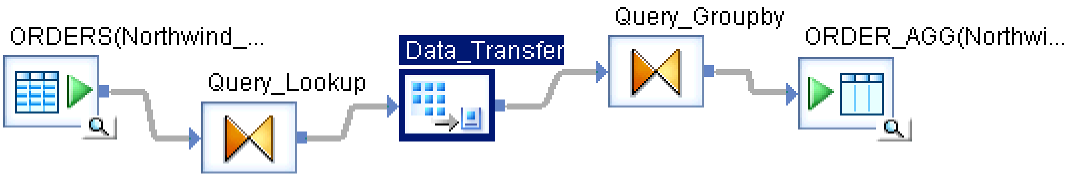
3.5.3.1.5 Example

This simple data flow contains a Query transform that does a lookup of sales subtotals and groups the results by country and region.



Suppose the GROUP BY operation processes millions of rows. Data Services cannot push the GROUP BY operation down to the database because the Query before it contains a **<lookup_ext>** function which Data Services cannot push down. You can add a **<Data_Transfer>** transform to split the **<lookup_ext>** function and the GROUP BY operation into two sub data flows to enable Data Services to push the GROUP BY to the target database.

1. Delete the connecting line between the two Query transforms.
2. Drag the Data_Transfer transform from the object library to the data flow in the work space and connect it to the two Query transforms.



3. Open the Data_Transfer transform editor and select the transfer type.

   This example selects Table for Transfer type.
4. In the Table options area, click the browse button for *Table name* and double-click the datastore that you want to contain the data to transfer to the second sub data flow.
5. Specify the transfer table in the Input table for Data_Transfer window.
   a) Enter the name of the table that will contain the data to transfer. The table does not need to exist.
   b) In the *Owner name* text box, type either the owner name or an alias.
   If you specify an alias for the owner, the *CREATE TABLE* statement in the *Options* tab shows the alias name in square brackets. When you execute the job, Data Services replaces the alias name with the owner name.
   c) Click *OK*.
6. You can change the default value for *Array fetch size*.
7. When you execute the job, Data Services displays messages for each sub data flow. For the sample GROUP BY data flow, the following messages display for the data flow and sub data flows:

   Job **<LookupGroup_Orders_Job>** is started. Process to execute data flow**<LookupGroup_Orders_DF>** is started. Starting sub data flow **<LookupGroup_Orders_DF_1>** on job server host <SJ-Comput>, port <3508>. Distribution level <Job>. Process to execute sub data flow **<LookupGroup_Orders_DF_1>** is started. Sub data flow **<LookupGroup_Orders_DF_1>** is started. Cache statistics for sub data flow

<<**LookupGroup_Orders_DF_1>** are not available to be used for optimization and need to be collected before they can be used. Sub data flow **<LookupGroup_Orders_DF_1>** using PAGEABLE Cache with <1280 MB> buffer pool. Sub data flow **<LookupGroup_Orders_DF_1>** is completed successfully. Process to execute sub data flow **<LookupGroup_Orders_DF_1>** is completed. Starting sub data flow **<LookupGroup_Orders_DF_2>** on job server host <SJ-Comput>, port <3508>. Distribution level <Job>. Process to execute sub data flow **<LookupGroup_Orders_DF_2>** is started. Sub data flow **<LookupGroup_Orders_DF_2>** is started. Cache statistics determined that sub data flow <LookupGroup_Orders_DF_2> uses <0> caches with a total size of <0> bytes. This is less than (or equal to) the virtual memory <1610612736> bytes available for caches. Statistics is switching the cache type to IN MEMORY. Sub data flow **<LookupGroup_Orders_DF_2>** using IN MEMORY Cache. Sub data flow **<LookupGroup_Orders_DF_2>** is completed successfully. Process to execute sub data flow **<LookupGroup_Orders_DF_2>** is completed. Process to execute data flow **<LookupGroup_Orders_DF>** is completed. Job **<LookupGroup_Orders_Job>** is completed successfully.

3.5.3.2 Date_Generation

Produces a series of dates incremented as you specify.

Use this transform to produce the key values for a time dimension target. From this generated sequence you can populate other fields in the time dimension (such as day_of_week) using functions in a query.

3.5.3.2.1 Data inputs

None.

3.5.3.2.2 Options

Option	Description
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. Options are:

Option	Description
	- *Yes*: The source is always cached unless it is the outer-most source in a join. - *No*: The source is never cached. The default is *Yes*. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. For new jobs, specify the cache only in the Query transform editor.
End date	The last date in the output sequence. Use the same format used for *Start date* to specify this date.
Increment	The interval between dates in the output sequence. Select *Daily*, *Monthly*, or *Weekly*.
Join rank	Indicates the rank of the output data set relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. For new jobs, specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.
Start date	The first date in the output sequence. Specify this date using the following format: YYYY.MM.DD where `YYYY` is a year value, `MM` is a month value, and `DD` is a day value.

3.5.3.2.3 Data outputs

A data set with a single column named `DI_GENERATED_DATE` containing the date sequence. The rows generated are flagged as INSERT.

The Date_Generation transform does not generate hierarchical data.

Generated dates can range from 1900.01.01 through 9999.12.31.

3.5.3.2.4 Example

To create a time dimension target with dates from the beginning of the year 1997 to the end of the year 2000, place a Date_Generation transform, a query, and a target in a data flow. Connect the output of the

Date_Generation transform to the query, and the output of the query to the target. Inside the Date_Generation transform, specify the following Options. You can also specify a variable for these options.

- *Start date*: 1997.01.01 (A variable can also be used.)
- *End date*: 2000.12.31 (A variable can also be used.)
- *Increment*: Daily (A variable can also be used.)

Inside the query, create two target columns and the field name, and define a mapping for these time dimension values:

- Business quarter: `BusQuarter`
  Function: quarter(Generated_date)
- Date number from start: `DateNum`
  Function: julian(generated_date) - julian(1997.01.01)

3.5.3.3 Effective_Date

Calculates an "effective-to" value for data that contains an effective date. The calculated effective-to date and an existing effective date produce a date range that allows queries based on effective dates to produce meaningful results.

3.5.3.3.1 Data inputs

Data that has an effective date column.

Effective dates allow you to indicate changes to information over time. The effective date value in each row of a data set indicates the date from which the data in the row is valid. As changes are made to the information, more rows are included to describe the information as it changes over time. Each row describing the set of information is distinguished from the others by the effective date of the row.

An example input data set might include a column that identifies the information being described (Project), a column that changes over time (Status), and an effective date:

Project	Effective date	Status
Cherry Lake	1999.06.22	Proposal
Cherry Lake	2003.01.12	Case
Hetch Hetchy Reservoir	1999.08.02	Proposal
Hetch Hetchy Reservoir	2003.05.06	Case

This transform description uses the term "related rows" to refer to a set of rows that describe the same information as it changes over time. There are two sets of related rows in the example above, described by the values in the Project column.

If the input data set allows duplicate effective dates, it might contain an effective sequence column to distinguish between related rows that also have the same effective date:

Project	Effective date	Effective sequence	Status
Cherry Lake	1999.06.22	0	Proposal
Cherry Lake	2002.01.12	0	Case
Hetch Hetchy Reservoir	1999.08.02	0	Proposal
Hetch Hetchy Reservoir	2002.10.17	0[a]	Proposal
Hetch Hetchy Reservoir	2002.10.17	1[a]	Case
Hetch Hetchy Reservoir	2003.05.06	0	Case
a. Project statuses are distinguished by the effective sequence.			

In the example, only the row with the largest sequence number is effective-dated by this transform. A query that selects the status of project "Hetch Hetchy Reservoir" on 2002.12.31 will return 'case' as a result. The input data set can contain only rows flagged as NORMAL.

The input data set can contain hierarchical data. The transform operates only on the rows at the top-level of the input data set, and passes nested data through to the output without change. Columns containing nested schemas cannot be used as transform parameters.

3.5.3.3.2 Editor

The Effective_Date transform editor includes:

- A Schema In pane on the left that shows the source schema
- A Schema Out pane on the right that shows the target schema
- An Effective Date tab that shows the transform options. You can drag column names from the source schema to fill in values for the *Effective Date column* and *Effective sequence column* options.

The target schema is generated in response to the values you choose in the transform options. To refresh the target schema after you make a change to the options, choose *View* > *Refresh* or press *F5*.

3.5.3.3.3 Options

Option	Description
Default effective to date value	A date assigned as the effective-to date for those rows with the highest effective date among related rows. You can also specify a variable for this option.
Effective date column	A column in the input data set of type `date` that contains the effective date. This column name is entered automatically if Data Services finds a column named `EFFDT` in the source. The column appears in the output schema with the name `EFFDT`. This field is required.
Effective sequence column	A column in the input data set that indicates the order in time of related rows that have duplicate effective dates. If no related rows also share effective dates, the sequence numbers are the same ('0' for example). If related rows do share the same effective dates, the sequence numbers are incremented as rows with conflicting effective dates are added. This transform returns only the row containing the maximum sequence number if there are related rows with the same effective date. This field is required only if the input data set allows duplicate effective dates.
Effective to column	The name of a date column added to the output schema that contains the effective-to date. The effective-to date for a row is equal to the effective date of the related row with the closest greater effective date. If no such row exists, the *Default effective to date* is used.

3.5.3.3.4 Data outputs

The transform output includes all of the columns from the source schema and the calculated effective-to date column. For example, given a default effective-to date of January 1, 2999, the input described in the data input section is transformed as follows:

Project	Effective date	Effective-to date	Status
Cherry Lake	1999.06.22	2003.01.12	Proposal
Cherry Lake	2003.01.12	2999.01.01[a]	Case
Hetch Hetchy Reservoir	1999.08.02	2002.10.17	Proposal

Project	Effective date	Effective-to date	Status
Hetch Hetchy Reservoir	2002.10.17	2003.05.06	Case
Hetch Hetchy Reservoir	2003.05.06	2999.01.01[a]	Case

a. The default effective-to date is used to close the effective date range.

In the case where an effective sequence column is necessary to produce a unique key—related rows contain the same effective date—the output from the Effective_Date transform includes a single row where the input had more than one. The row returned contains the largest sequence number:

Project	Effective date	Effective to date	Effective sequence	Status
Cherry Lake	1999.06.22	2003.01.12	0	Proposal
Cherry Lake	2003.01.12	2999.01.01	0	Case
Hetch Hetchy Reservoir	1999.08.02	2002.10.17	0	Proposal
Hetch Hetchy Reservoir	2002.10.17	2003.05.06	1[a]	Case
Hetch Hetchy Reservoir	2003.05.06	2999.01.01	0	Case

a. Data from the row with sequence 0 is omitted.

After the range of effective dates is generated for a set of data, you can use the effective-to date to filter appropriate records. For example, you can extract the subset of records valid as of today by selecting only those records whose effective-to column is later than today's date and effective-from column is earlier than today's date.

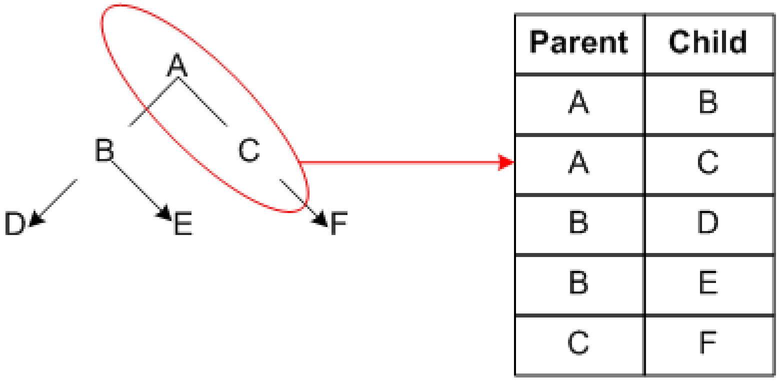
Nested schemas in the input are passed through without change.

3.5.3.4 Hierarchy_Flattening

Constructs a complete hierarchy from parent/child relationships, then produces a description of the hierarchy in vertically or horizontally flattened format.

3.5.3.4.1 Data inputs

Rows describing individual parent-child relationships.



Parent	Child
A	B
A	C
B	D
B	E
C	F

Each row must contain two columns that function as the keys of the parent and child in the relationship. The input can also include columns containing attributes describing the parent and/or child.

An example of an input data set is an Employee Master hierarchy description in which each row represents the relationship between an employee (child node) and the employee's manager (parent node). In this example:

- Employ_ID is the child node identifier and the primary key
- Mgr_ID is the parent node identifier

Employee
Employ_ID
Dept
Salary
Mgr_ID

The input data set cannot include rows with operation codes other than NORMAL.

The input data set can contain hierarchical data.

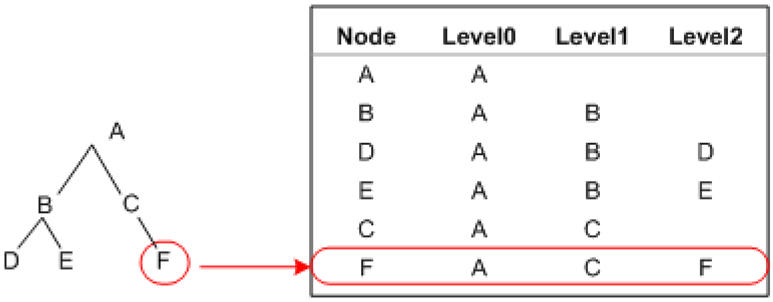
3.5.3.4.2 Editor

The Hierarchy_Flattening editor includes:

- Schema In pane which contains the source schema
- Schema Out pane which contains the target schema
- Hierarchy_Flattening transform options
  You can drag column names from the source schema to fill in values for the Parent column, Child column, Parent attribute list, and Child attribute list options.
  Data Services generates the target schema in response to the values you choose in the transform options. To refresh the target schema after you make a change to the options, choose *View* > *Refresh* or press *F5*.

3.5.3.4.3 Options

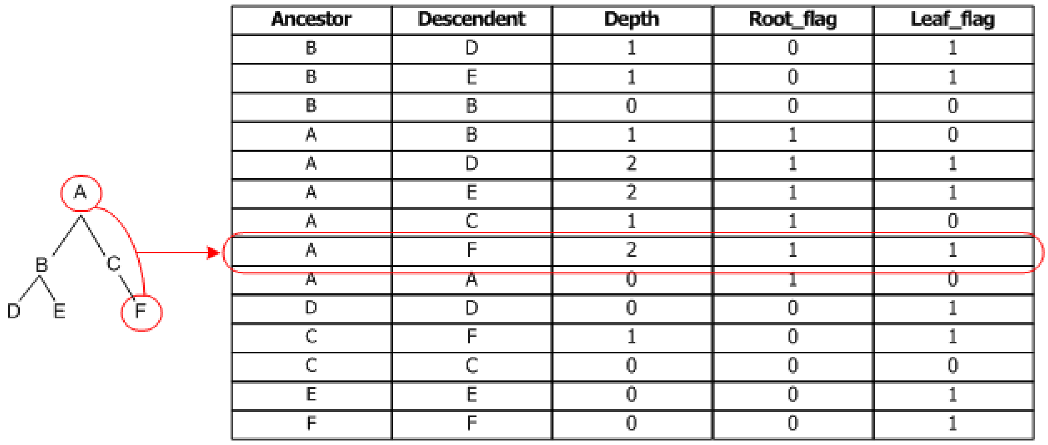
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td><i>Child attribute list</i></td><td>Identifies a column or columns that are associated with the child column. You can drag columns from the source schema into the <i>Child attribute list</i>. The column name appears in the target schema with a prefix that identifies the column as a child attribute. The following table shows the result of adding a column named <code>POPULATION</code> to the child attribute list.
<table>
<tr><th>Flattening mode</th><th>Source column</th><th>Target column</th></tr>
<tr><td>Vertical</td><td>POPULATION</td><td>C_POPULATION</td></tr>
<tr><td>Horizontal</td><td>POPULATION</td><td>C_L1_POPULATION
C_L2_POPULATION
(one column for each hierarchy level)</td></tr>
</table>
You can specify columns including nested schemas as the child attribute.</td></tr>
<tr><td><i>Child column</i></td><td>Identifies the column in the source data that contains the child identifier in each parent-child relationship. You can drag this column from the source schema into the <i>Child column</i> box.
You cannot specify columns including nested schemas as the child.</td></tr>
<tr><td><i>Do not abort in case of cycle</i></td><td>Select to specify that a job should not abort if a cycle (circular dependency) is detected. If a cycle is encountered, warnings are written to the log files.
If left unchecked, jobs will abort if the transform encounters a cycle.</td></tr>
<tr><td><i>Flattening type</i></td><td>Indicates how the hierarchical relationships are described in the output. Choose from two options:
<ul>
<li><i>Horizontal</i> — Each row of the output describes a single node in the hierarchy and the path to that node from the root. This mode requires that you specify the maximum path length through the tree as the <i>Maximum depth</i>.



<table>
<tr><th>Node</th><th>Level0</th><th>Level1</th><th>Level2</th></tr>
<tr><td>A</td><td>A</td><td></td><td></td></tr>
<tr><td>B</td><td>A</td><td>B</td><td></td></tr>
<tr><td>D</td><td>A</td><td>B</td><td>D</td></tr>
<tr><td>E</td><td>A</td><td>B</td><td>E</td></tr>
<tr><td>C</td><td>A</td><td>C</td><td></td></tr>
<tr><td>F</td><td>A</td><td>C</td><td>F</td></tr>
</table>
</li>
<li><i>Vertical</i> — Each row of the output describes a single relationship between ancestor and descendent and the number of nodes the relationship includes. There is a row in the output for</li>
</ul></td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td>

each node and all of the descendents of that node. Each node is considered its own descendent and therefore is listed one time as both ancestor and descendent.



<table>
<tr><th>Ancestor</th><th>Descendent</th><th>Depth</th><th>Root_flag</th><th>Leaf_flag</th></tr>
<tr><td>B</td><td>D</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>B</td><td>E</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>B</td><td>B</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>A</td><td>B</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>A</td><td>D</td><td>2</td><td>1</td><td>1</td></tr>
<tr><td>A</td><td>E</td><td>2</td><td>1</td><td>1</td></tr>
<tr><td>A</td><td>C</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>A</td><td>F</td><td>2</td><td>1</td><td>1</td></tr>
<tr><td>A</td><td>A</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>D</td><td>D</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>C</td><td>F</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>C</td><td>C</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>E</td><td>E</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>F</td><td>F</td><td>0</td><td>0</td><td>1</td></tr>
</table>

</td></tr>
<tr><td><i>Generate cycle rows</i></td><td>

Specifies that when a cycle is encountered, the circular node will be the last in the hierarchy tree, and the tree itself will carry negative value for Leave Level (Horizontal Flattening) and Depth (Vertical Flattening).

Select this option if you would like more information about which nodes are causing the cycles. For example, you can insert a Validation transform after the Hierarchy flattening transform to check for negative values. If negative values are encountered, you can send the the data to another path for further analysis.

</td></tr>
<tr><td><i>Maximum depth</i></td><td>

(This option only applies to horizontal flattening.) Indicates the maximum depth of the hierarchy. The root node (level 0) has a depth of 0; the first level has a depth of 1, and so on.

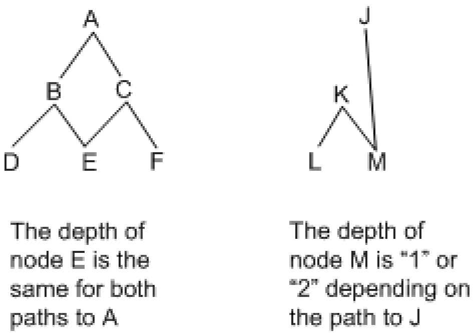
If you do not know the number of levels in your hierarchy, set <i>Maximum depth</i> to 1. When you execute the job, a warning message will appear in the execution log indicating that the <i>Maximum depth</i> is less than the actual depth of the hierarchy. Reset <i>Maximum depth</i> to the actual value reported in the warning message.

</td></tr>
<tr><td><i>Parent attribute list</i></td><td>

Identifies a column or columns that are associated with the parent column. You can drag columns from the source schema into the <i>Parent attribute list</i>. The column name appears in the target schema with a prefix that identifies the column as a parent attribute. The following table shows the result of adding a column named `POPULATION` to the parent attribute list.

<table>
<tr><th>Flattening mode</th><th>Source column</th><th>Target column</th></tr>
<tr><td>Vertical</td><td>POPULATION</td><td>P_POPULATION</td></tr>
<tr><td>Horizontal</td><td>POPULATION</td><td>P_L1_POPULATION
P_L2_POPULATION</td></tr>
</table>

</td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><table><tr><th>Flattening mode</th><th>Source column</th><th>Target column</th></tr><tr><td></td><td></td><td>(one column for each hierarchy level)</td></tr></table>You can specify columns including nested schemas as the parent attribute.</td></tr>
<tr><td><em>Parent column</em></td><td>Identifies the column in the source data that contains the parent identifier in each parent-child relationship. You can drag this column from the source schema into the <em>Parent column</em> box.
You cannot specify columns including nested schemas as the parent.</td></tr>
<tr><td><em>Run as separate process</em></td><td>This option creates a separate sub data flow process for the Hierarchy_Flattening transform when Data Services executes the data flow.</td></tr>
<tr><td><em>Use maximum length paths</em></td><td>(This option only applies to vertical flattening.) Indicates whether longest or shortest paths are used to describe relationships between descendents and ancestors when the descendent has more than one parent. The option only affects the DEPTH column in the output.

The depth of node E is the same for both paths to A
The depth of node M is "1" or "2" depending on the path to J</td></tr>
</table>

Related Information

Performance Optimization Guide: Run as a separate process option [page 2153]

3.5.3.4.4 Data outputs

Horizontal flattening represents each level in the hierarchy as a column in the output, with the root listed in the first column and the outermost leaf listed in the last column. Represented horizontally, the number of levels in the hierarchy and the distance between a given node and the root node is clear.

Horizontal flattening produces the following target columns:

Column name	Description
CURRENT_LEAF	The end node described.
LEAF_LEVEL	The number of levels down from the root node where Current_leaf is found The root node has Leaf_level of 0.
LEVEL0	The descriptor for the top level node.
LEVEL1	The descriptor for the first level node If Leaf_level is 0, this value is NULL.
LEVEL **<n>**	The descriptor for the **<n>** th level node where **<n>** is the number of levels in the hierarchy If Leaf_level is **<n>** -1 or less, this value is NULL.
P_L0_ **<attribute_column>**	Attribute column associated with the node described in Level0.
C_L1_ **<attribute_column>**	Attribute column associated with the node described in Level1 when that node is the child node. If Leaf_level is 0, this value is NULL.
P_L1_ **<attribute_column>**	Attribute column associated with the node described in Level1 when that node is the parent node. If Leaf_level is 0, this value is NULL.
C_L **<n-1>** _ **<attribute_column>**	Attribute column associated with the child node described in Level **<n-1>** where **<n>** is the number of levels in the hierarchy. If Leaf_level is **<n>** -2, this value is NULL.
P_L **<n-1>** _ **<attribute_column>**	Attribute column associated with the child node described in Level **<n-1>** where **<n>** is the number of levels in the hierarchy. If Leaf_level is **<n>** -2, this value is NULL.
C_L **<n>** _ **<attribute_column>**	Attribute column associated with the child node described in Level **<n>** where **<n>** is the number of levels in the hierarchy. If Leaf_level is **<n>** -1, this value is NULL.

The following table shows an example of the target schema and data for horizontal flattening with a two-level hierarchy including country as the root node, state at level one, and city at level 2 (leaf nodes). The parent and child attributes are the same, a population value.

The table shows the three rows in the output; however, the format folds the row data onto two rows. The headings at the top of the table describe the first half of each row of data. The headings at the bottom of the table describe the second half of each row of data.

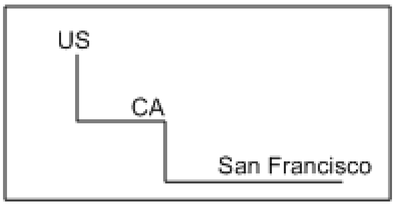
CURRENT_LEAF	LEAF_LEVEL		LEVEL0		LEVEL1		LEVEL2	
US	0		US		NULL		NULL	
		272,583,805		NULL		NULL		NULL
CA	1		US		CA		NULL	
		272,583,805		30,866,851		NULL		NULL
San Francisco	2		US		CA		San Francisco	
		272,583,805		30,866,851		30,866,851		723,959
		P_L0_POP	C_L1_POP		P_L1_POP		C_L2_POP	

In a typical data flow including a Hierarchy_Flattening transform with the same attribute for parent and child nodes, you may follow the transform with a query that would filter the duplicated attribute values. The query would pass the P_L0_ `<attribute_column>` and the C_L `<n>` _ `<attribute_column>` through as they are, then keep either parent or child attribute for the intermediate levels.

Vertical flattening produces the following target columns:

Column name	Description
ANCESTOR	The node closer to the root node in the relationship described by this row.
DESCENDENT	The node farther from the root node in the relationship described by this row.
DEPTH	Number of levels between the Ancestor and Descendent.
ROOT_FLAG	Identifies the value in the Ancestor column as the top node of the hierarchy. Root_flag is 1 if Ancestor is the root node. Otherwise, Root_flag is 0.
LEAF_FLAG	Identifies the value in the Descendent column as the bottom node of the hierarchy. Leaf_flag is 1 if Descendent is the leaf node. Otherwise, Leaf_flag is 0.
P_`<attribute_column>`	Column from the source that you associate with the parent (can be more than one P_ `<attribute_column>`). If Leaf_flag is 1, this value is NULL.
C_`<attribute_column>`	Column from the source that you associate with the child (can be more than one C_ `<attribute_column>`). If Root_flag is 1, this value is NULL.

The following graphic and table shows an example of the target schema and data for vertical flattening with a two-level hierarchy. The hierarchy includes a country as the root node, state at level one, and city at level 2 (leaf nodes).



The parent and child attributes are the same, a population value.

ANCESTOR	DESCENDENT	DEPTH	ROOT _FLAG	LEAF _FLAG	P_POP	C_POP
US	CA	1	1	0	272,583,805	30,866,851
US	SanFrancisco	2	1	1	272,583,805	723,959
CA	SanFrancisco	1	0	1	30,866,851	723,959
US	US	0	1	0	272,583,805	NULL
CA	CA	0	0	0	30,866,851	30,866,851
SanFrancisco	SanFrancisco	0	0	1	NULL	723,959

Each node is listed one time as both ancestor and descendent. The Parent attribute is null for a row describing the relationship between a leaf node and itself. Likewise the Child attribute is null for a row describing the relationship between a root node and itself.

The transform ignores any hierarchical data unless a nested schema is identified as a parent or child attribute. An attribute column containing nested data is passed through the transform without change.

3.5.3.4.5 Error conditions

If the hierarchy represented by the input data set is cyclic—some node is its own ancestor—Data Services produces a run-time error.

> i Note
>
> This runtime error does not occur if you select the *Do not abort in case of cycle* option.

No errors are produced if the input data source describes multiple root nodes.

3.5.3.5 History_Preserving

The History_Preserving transform allows you to produce a new row in your target rather than updating an existing row. You can indicate in which columns the transform identifies changes to be preserved.

If the value of certain columns change, this transform creates a new row for each row flagged as UPDATE in the input data set.

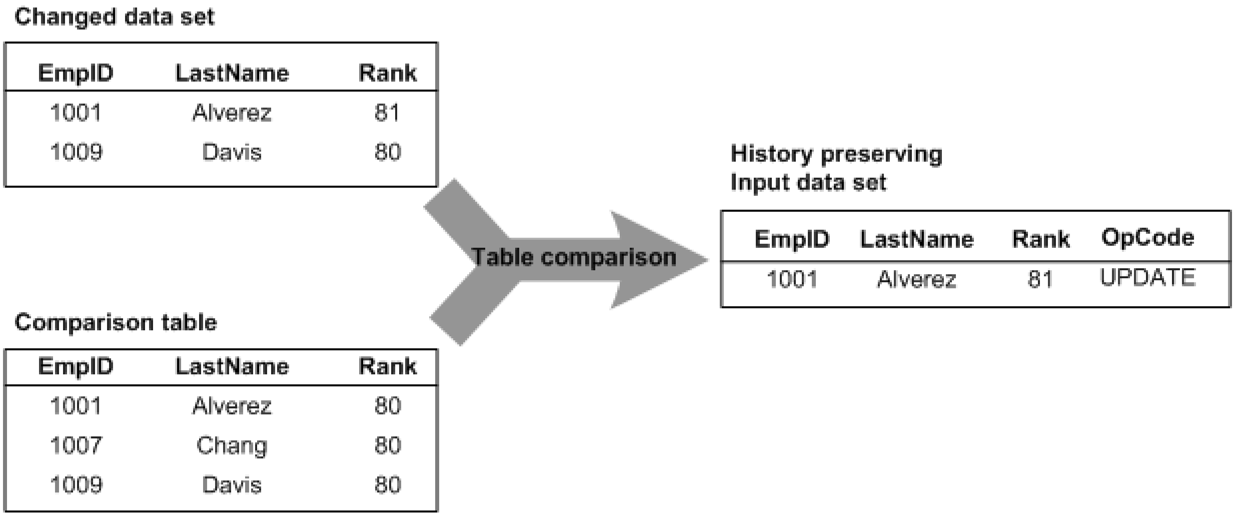
Related Information

Table_Comparison [page 1110]
real [page 1036]

3.5.3.5.1 Data inputs

A data set that is the result of a comparison between two images of the same data in which changed data from the newer image are flagged as UPDATE rows and new data from the newer image are flagged as INSERT rows.

For example, a target table that contains employee job rankings information is updated periodically from a source table. In this case, the table comparison flags changed data for employee Alverez and discards unchanged data for employee Davis. The result is a single row flagged with the UPDATE operation code.



Changed data set

EmpID	LastName	Rank
1001	Alverez	81
1009	Davis	80

Comparison table

EmpID	LastName	Rank
1001	Alverez	80
1007	Chang	80
1009	Davis	80

History preserving
Input data set

EmpID	LastName	Rank	OpCode
1001	Alverez	81	UPDATE

The input data set can contain hierarchical data. The transform operates only on the rows at the top-level of the input data set, and passes nested data through to the output without change. Columns containing nested schemas cannot be used as transform parameters.

Use caution when using columns of data type `real` in this transform. Comparison results are unpredictable for this data type.

3.5.3.5.2 Editor

The History_Preserving transform editor includes the source schema, the target schema, and transform options. You can drag column names from the source schema to fill in values for the Date columns *Valid from* and *Valid to* date column, *Compare columns*, and Current flag *Column* options. Data Services generates the target schema in response to the values you choose in the transform options. To refresh the target schema after you make a change to the options, choose ▶ *View* ▶ *Refresh* ◀ or press *F5*.

3.5.3.5.3 Options

Option	Description
Compare columns	The column or columns in the input data set for which this transform compares the before- and after-images to determine if there are changes. ● If the values in each image of the data match, the transform flags the row as UPDATE. The result updates the warehouse row with values from the new row. The row from the before-image is included in the output as UPDATE to effectively update the date and flag information. ● If the values in each image do not match, the row from the after-image is included in the output of the transform flagged as INSERT. The result adds a new row to the warehouse with the values from the new row. ● *Compare columns* cannot contain nested schemas.
Current flag—Column	A column from the source schema that identifies the current valid row from a set of rows with the same primary key. You can indicate whether a row is the most current data in the warehouse for a given primary key using this flag. The current flag *Column* cannot be the same value as the *Valid from* or the *Valid to* date column. The current flag *Column* cannot contain a nested schema. Data Services validates that you specify a current flag *Column*, *Set value*, and *Reset value*, if any are specified.
Current flag—Reset value	An expression that evaluates to a value with the same data type as current flag *Column*. Data Services uses this value to update the current flag *Column* in an existing row in the warehouse that included changes in one or more of the compare columns. Enter a value in the current flag *Column* box to enable the *Flag reset value*. For added flexibility, you can enter a variable for this option.
Current flag—Set value	An expression that evaluates to a value with the same data type as the current flag *Column*. Data Services uses this value to update the current flag *Column* in the new row in the warehouse added to preserve history of an existing row. Enter a value in the Current flag *Column* box to enable the *Set value*. For added flexibility, you can enter a variable for this option.
Date columns—Valid from	A date or datetime column from the source schema. If the warehouse uses an effective date to track changes in data, specify a *Valid from* date column. Data Serv-

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td>ices uses this value in the new row in the warehouse added to preserve the history of an existing row. Data Services also uses this value to update the *Valid to* date column in the previously current row in the warehouse.
Data Services validates that both the *Valid from* date column and the *Valid to* date column have been specified if either is specified.</td></tr>
<tr><td>*Date columns—Valid to*</td><td>A date or datetime column from the source schema. Specify a if the warehouse uses an effective date to track changes in data and if you specified a *Valid from* date column.
This value is used as the new value in the *Valid to* date column in the new row added to the warehouse to preserve history of an existing row.
The *Valid to* date column cannot be the same as the *Valid from* date column.</td></tr>
<tr><td>*Date columns—Valid to date value*</td><td>Specify the values to use in the *Valid to* date column in the old record and the new record added to the warehouse to preserve history of an existing row.
● *New record*—You can specify one of the following values:
○ A date value specified as a four-digit year followed by a period, followed by a two-digit month, followed by a period, and followed by a two-digit day value. The default value is `9000.12.31`. A variable can also be used.
○ A variable name that contains a date value.
● *Old record*—You can specify one of the following values:
○ *Use "valid from" date of new record*—The following example shows that the new record (Key 2) column *From_Date* contains `2006.01.31` and the old record (Key 1) column *To_Date* contains this same value.
<table>
<tr><th>Key</th><th>Empno</th><th>Name</th><th>Salary</th><th>From_Date</th><th>To_Date</th></tr>
<tr><td>1</td><td>100</td><td>Chang</td><td>10000.00</td><td>2006.01.01</td><td>2006.01.31</td></tr>
<tr><td>2</td><td>100</td><td>Chang</td><td>20000.00</td><td>2006.01.31</td><td>9000.12.31</td></tr>
</table>
○ *Use one day before "valid from" date of new record*—The following example shows that the new record (Key 2) column *From_Date* contains `2006.01.31` and the old record (Key1) column *To_Date* contains a date that is one day before that value, `2006.01.30`.
<table>
<tr><th>Key</th><th>Empno</th><th>Name</th><th>Salary</th><th>From_Date</th><th>To_Date</th></tr>
<tr><td>1</td><td>100</td><td>Chang</td><td>10000.00</td><td>2006.01.01</td><td>2006.01.30</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><table><tr><th>Key</th><th>Empno</th><th>Name</th><th>Salary</th><th>From_Date</th><th>To_Date</th></tr><tr><td>2</td><td>100</td><td>Chang</td><td>20000.00</td><td>2006.01.31</td><td>9000.12.31</td></tr></table></td></tr>
<tr><td><i>Preserve delete row(s) as update row(s)</i></td><td>Converts DELETE rows to UPDATE rows in the target warehouse and, if you previously set effective date values (<i>Valid from</i> and <i>Valid to</i>), sets the <i>Valid To value</i> to the execution date. Use this option to maintain slowly changing dimensions by feeding a complete data set first through the Table Comparison transform with its <i>Detect deleted row(s) from comparison table</i> option selected.</td></tr>
</table>

3.5.3.5.4 Data outputs

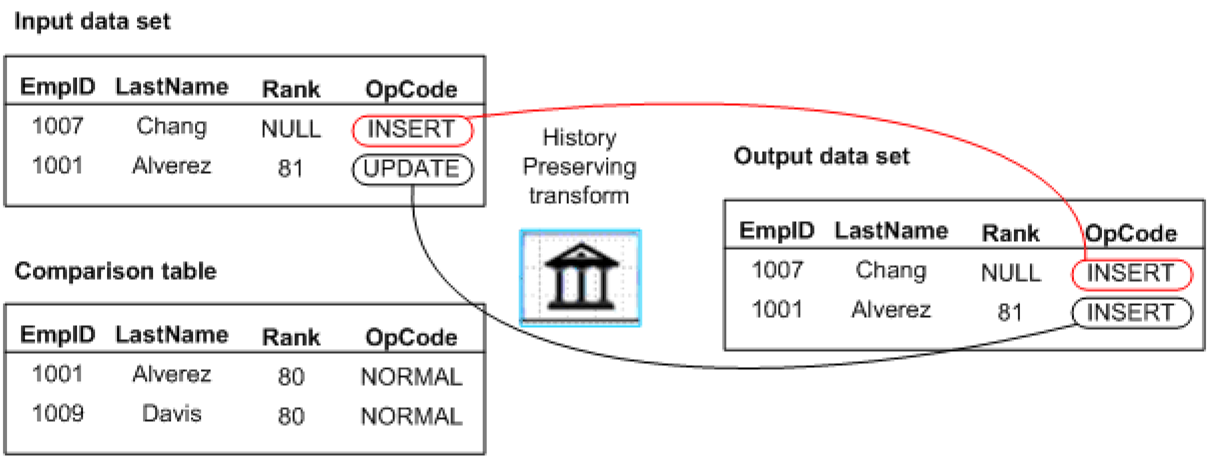
Data Outputs

A data set with rows flagged as INSERT or UPDATE.

For each row in the input data set, there are two possible outcomes from the transform:

- An INSERT row
  A new row must be added to the comparison table if:
  - Given an input row flagged as UPDATE—A value in a compare column from the input data set does not match a corresponding value in the comparison table.
  - Given an input row flagged as INSERT—The primary key from the input data set does not appear in the comparison table.

  The transform produces an INSERT row with the values from the input data set row.



Input data set

EmpID	LastName	Rank	OpCode
1007	Chang	NULL	INSERT
1001	Alverez	81	UPDATE

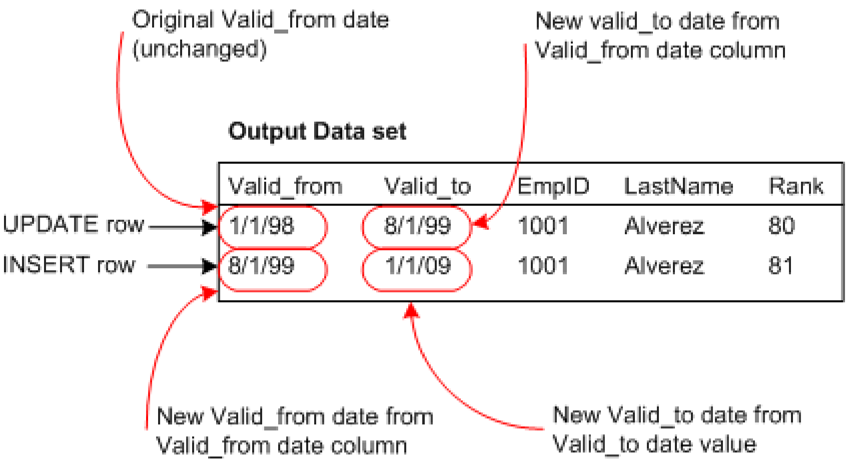
Comparison table

EmpID	LastName	Rank	OpCode
1001	Alverez	80	NORMAL
1009	Davis	80	NORMAL

Output data set

EmpID	LastName	Rank	OpCode
1007	Chang	NULL	INSERT
1001	Alverez	81	INSERT

If you specified flag values for the History_Preserving transform, Data Services includes the Flag set value in the INSERT row. In addition Data Services includes an UPDATE row to update the previously current row in the warehouse with the Flag reset value.
If you specified effective date columns (Valid to date column and Valid from date column), Data Services includes this data as well.



Valid_from	Valid_to	EmpID	LastName	Rank
1/1/98	8/1/99	1001	Alverez	80
8/1/99	1/1/09	1001	Alverez	81

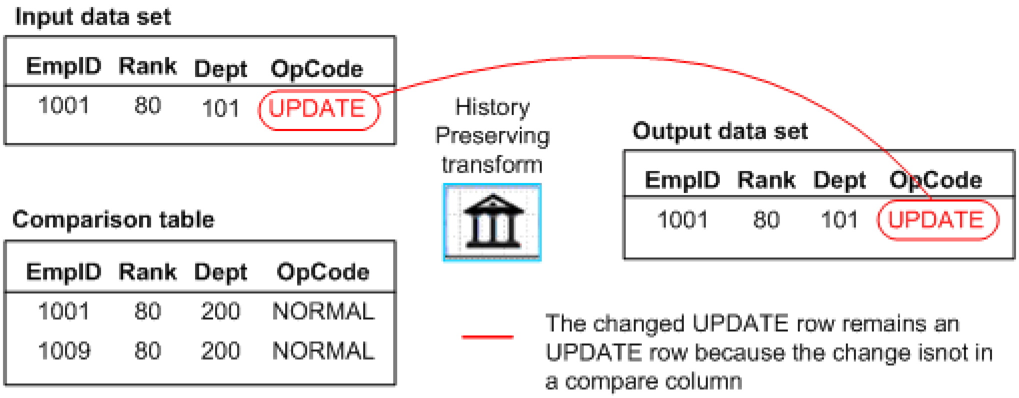
Original Valid_from date (unchanged)

New valid_to date from Valid_from date column

New Valid_from date from Valid_from date column

New Valid_to date from Valid_to date value

- An UPDATE row
  Input rows flagged as UPDATE contain changes, but not in the compare columns. The transform produces an UPDATE row with the values from the input data set row.



Input data set

EmpID	Rank	Dept	OpCode
1001	80	101	UPDATE

Comparison table

EmpID	Rank	Dept	OpCode
1001	80	200	NORMAL
1009	80	200	NORMAL

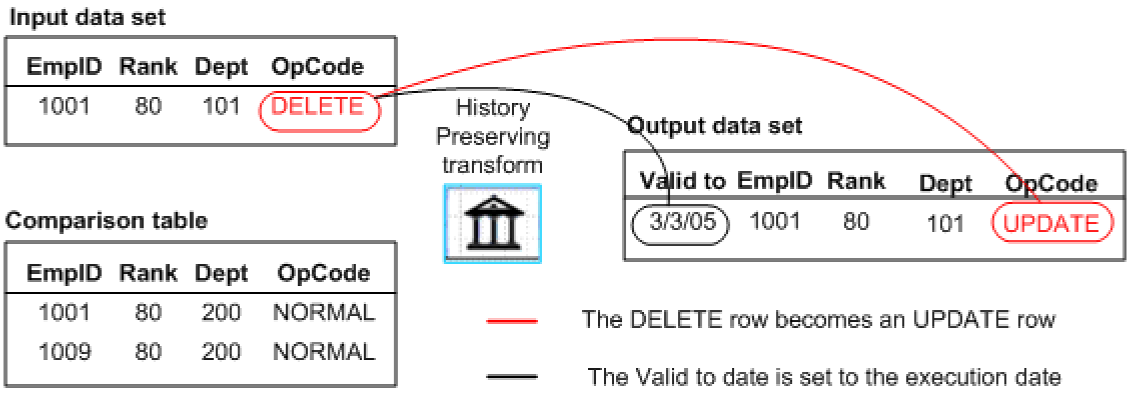
Output data set

EmpID	Rank	Dept	OpCode
1001	80	101	UPDATE

The changed UPDATE row remains an UPDATE row because the change isnot in a compare column

Input rows flagged as DELETE contain changes, but not in the compare columns. The transform produces an UPDATE row with the values from the input data set row. If you specified the effective date column *Valid to* in the History Preserving transform, Data Services sets its value to the execution date.

Input rows flagged as DELETE contain changes, but not in the compare columns. The transform produces an UPDATE row with the values from the input data set row.



Input data set

EmpID	Rank	Dept	OpCode
1001	80	101	DELETE

Comparison table

EmpID	Rank	Dept	OpCode
1001	80	200	NORMAL
1009	80	200	NORMAL

Output data set

Valid to	EmpID	Rank	Dept	OpCode
3/3/05	1001	80	101	UPDATE

History Preserving transform
The DELETE row becomes an UPDATE row
The Valid to date is set to the execution date

Nested schemas in the input are passed through without change.

3.5.3.6 Key_Generation

Generates new keys for new rows in a data set.

When it is necessary to generate artificial keys in a table, the Key_Generation transform looks up the maximum existing key value from a table and uses it as the starting value to generate new keys. The transform expects the generated key column to be part of the input schema.

3.5.3.6.1 Data inputs

A data set that is the result of a comparison between two images of the same data in which changed data from the newer image are flagged as UPDATE rows and new data from the newer image are flagged as INSERT rows.

The data set includes a column into which generated keys are added.

The input data set can contain hierarchical data. The transform operates only on the rows at the top-level of the input data set, and passes nested data through to the output without change. Columns containing nested schemas cannot be used as transform parameters.

3.5.3.6.2 Editor

The Key_Generation transform editor includes the source schema, the target schema, and transform options. To refresh the target schema after you make a change to the options, choose *View* > *Refresh* or press *F5*.

3.5.3.6.3 Options

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td><i>Generated key column</i></td><td>The column in the key source table containing the existing keys values. A column with the same name must exist in the input data set; the new keys are inserted in this column.</td></tr>
<tr><td><i>Increment value</i></td><td>The interval between generated key values. For added flexibility, you can enter a variable for this option.</td></tr>
<tr><td><i>Table name</i></td><td>The fully qualified name of the source table from which the maximum existing key is determined (key source table). This table must be already imported into the repository. <i>Table name</i> is represented as "DATASTORE.OWNER.TABLE" where DATASTORE is the name of the datastore Data Services uses to access the key source table; OWNER, if required, depends on the database type associated with the table:
<table>
<tr><th>Database type</th><th>Owner value</th></tr>
<tr><td>DB2</td><td>Data source dependent</td></tr>
<tr><td>Informix</td><td>Informix-defined user name</td></tr>
<tr><td>Microsoft SQL Server</td><td>User name</td></tr>
<tr><td>ODBC</td><td>Data source dependent</td></tr>
<tr><td>Oracle</td><td>User name</td></tr>
<tr><td>SAP Sybase ASE</td><td>User name</td></tr>
<tr><td>SAP Sybase IQ</td><td>User name</td></tr>
</table>
</td></tr>
</table>

3.5.3.6.4 Data outputs

The input data set with the addition of key values in the generated key column, for input rows flagged as INSERT.

Use the Key_Generation transform to produce keys that distinguish rows that would otherwise have the same primary key. For example, suppose the History_Preserving transform produces rows to add to a warehouse and these rows have the same primary key as rows that already exist in the warehouse. In this case, you can add a generated key to the warehouse table and fill new key values using the Key_Generation transform.

3.5.3.7 Map_CDC_Operation

Using its input requirements (values for the *Sequencing column* and a *Row operation column*), performs three functions:

- Sorts input data based on values in *Sequencing column* drop-down list and (optional) the *Additional grouping columns* box.
- Maps output data based on values in *Row operation column* drop-down list. Source table rows are mapped to INSERT, UPDATE, or DELETE operations before passing them on to the target.
- Resolves missing, separated, or multiple before- and after-images for UPDATE rows.
- Allows you filter columns and view UPDATE rows prior to running the job.

While commonly used to support relational or mainframe changed-data capture (CDC), this transform supports any data stream as long as its input requirements are met. Relational CDC sources include Oracle and SQL Server.

This transform is typically the last object before the target in a data flow because it produces INPUT, UPDATE and DELETE operation codes. Data Services produces a warning if other objects are used.

Related Information

Designer Guide: Importing CDC data from Oracle [page 745]

3.5.3.7.1 Data inputs

All rows in the input data are set to NORMAL. This is an internal Data Services operation code.

The input data set can contain hierarchical data. Nested schemas in the input are passed through without change.

3.5.3.7.2 Editor

Use the drop-down lists in the *CDC Columns* section to select the required columns in your input schema used to sequence the input rows and map operations to the output.

If you want to sort the input using additional columns, click a column in the input schema and drag it into the *Additional grouping columns* box.

If you are using a relational or mainframe CDC source table, the `DI_SEQUENCE_NUMBER` and `DI_OPERATION_TYPE` columns appear in the input schema and are automatically removed in the output schema. To propagate these columns to the output, create additional columns that map to them prior to coming into the Map_CDC_Operation transform.

3.5.3.7.3 Options

Option	Description
Additional grouping columns	In addition to the *Sequencing column*, you can sort input on additional columns by dragging them into this box from the input schema. Sorts are prioritized based first on the sequencing column and then on the order of the columns added to this box.
Define columns to filter updated rows	Drag and drop columns from the input schema to this box to compare CDC updated rows in the before and after images. If the values in the columns differ between the before and after images, Data Services generates an UPDATE row, otherwise the rows are filtered out.
Input already sorted by sequencing column	This transform by default assumes that the input data is already sorted based on the value selected in the *Sequencing column* box. If you deselect this check box, Data Services will re-sort the input data using the value in the *Sequencing column* box. Use the re-sort capability of this transform only when necessary as it impacts job performance.
Row operation column	(Required) Specifies a column with one of the following output operation codes for each row: ● `I` for INSERT ● `B` for before-image of an UPDATE ● `U` for after-image of an UPDATE ● `D` for DELETE If you are using a relational or mainframe CDC source table, the `DI_OPERATION_TYPE` column is automatically selected as the *Row operation column*.
Sequencing column	(Required) Specifies an integer used to order table rows. If you are using a relational or mainframe CDC source table, the `DI_SEQUENCE_NUMBER` column is automatically selected as the *Sequencing column*.

3.5.3.7.4 Data outputs

A data set with rows flagged as specified by the values in the column selected as the *Row operation column*.

Rows in the input data set all use NORMAL as their internal Data Services operation code.

Rows in the output data set can have any of the following operation codes:

- INSERT
- DELETE
- UPDATE

In addition, the DISCARD option is assigned under some conditions. Discarded rows are not passed through to the output of the transform.

3.5.3.7.5 Sorting CDC data

When you apply changed data to a loader, it is important that the order of the rows is preserved. For example, if the following operations are applied to an empty target:

- INSERT into TAB1 values ('Bob', 'Boat', 3500)
- INSERT into TAB1 values ('Jane', 'BMW Roadster', 24000)
- UPDATE TAB1 set toy = 'Motorcycle', price = 12000 where name = 'Bob'
- DELETE from TAB1 where name = 'Bob'

The table TAB1 will be left with one row:

```
'Jane', 'BMW Roadster', 24000
```

If these operations are applied out of order, for example, if the DELETE occurs before the UPDATE operation, then database consistency is no longer preserved. In this example:

- The table has two rows (Bob and Jane)
- The last UPDATE statement fails because there is no row on which to perform an UPDATE

By ordering the input rows using the sequencing column, the order of the original set of operations is preserved.

The sequencing column values are also useful if you are using before- and after-images for update rows because it is possible that before- and after-image pairs may be separated, multiplied or lost depending on the design of your data flow. You can re-sort input columns as needed by using the sequencing column and any number of additional columns.

The before- and after-images of an UPDATE row have the same sequence value. Thus correctly sorted before- and after-image rows are listed in pairs.

3.5.3.7.6 Rules for resolving before- and after-image pairs

The Map_CDC_Operation transform uses the following rules to process and resolve before- and after-images:

- When constructing UPDATE rows, the value in the *Row Operation Column* is used. If there are before-images in the input stream, before- (`B`) and after-image (`U`) row pairs are combined into one UPDATE row.
  For example, given the following sample input of six rows:

Sequencing Column	Operation Column	Internal Operation Code
1	I	NORMAL
2	B	NORMAL
2	U	NORMAL
3	D	NORMAL

Sequencing Column	Operation Column	Internal Operation Code
4	B	NORMAL
4	U	NORMAL

The following four rows will be the output:

Sequencing Column	Operation Column	Internal Operation Code
1	I	INSERT
2	U	UPDATE (before- and after-images)
3	D	DELETE
4	U	UPDATE (before- and after-images)

- If there are no before-images (`B`) in the input stream, the after-images (`U`) alone produce UPDATE rows.
  Given the following sample input rows:

Sequencing Column	Operation Column	Internal Operation Code
1	I	NORMAL
2	U	NORMAL
3	D	NORMAL
4	U	NORMAL

The following four rows will be the output:

Sequencing Column	Operation Column	Internal Operation Code
1	I	INSERT
2	U (no before)	UPDATE
3	D	DELETE
4	U (no before)	UPDATE

- If a before-image (`B`) row is followed by additional `B` rows, the subsequent `B` rows are ignored until an after-image (`U`) row is encountered.
  For example, given the following six input rows:.

Sequencing Column	Operation Column	Internal Operation Code
1	U	NORMAL
1	B	NORMAL
2	B	NORMAL
3	B	NORMAL
4	B	NORMAL
2	U	NORMAL

Two UPDATE rows are output:

Sequencing Column	Operation Column	Internal Operation Code
1	U	UPDATE (before- and after-images)
2	U	UPDATE (before- and after-images)

- ○ The first two rows are processed as one UPDATE row.
- ○ The third and sixth row are processed as a pair. One UPDATE row is output.
- ○ The remaining rows are DISCARDED. For more information, see the following rule.

- If after a series of B rows, either no U rows remain or another row type is encountered, B rows are discarded. For example, given the following sample input of five rows:

Sequencing Column	Operation Column	Internal Operation Code
1	U	NORMAL
1	B	NORMAL
2	B	NORMAL
3	B	NORMAL
4	I	NORMAL

Two rows are output:

Sequencing Column	Operation Column	Internal Operation Code
1	U	UPDATE (before- and after-images)
4	I	INSERT

Related Information

Designer Guide: Using before-images [page 750]

3.5.3.7.7 Filtering updated rows

You can drag and drop columns from the input schema to the *Define columns to filter updated rows* box to filter CDC updated rows.

Filter criteria: If two input rows have the same sequencing column value and operation column value (B and U), the Map_CDC_Operation transform compares the before image and after image of the seleted columns in the column filter.

If the values in the columns in the filter differ between the before image and after image, Data Services generates an UPDATE row; otherwise, the row will be filtered out. If there is no column filter, all updated rows coming from the input stream will be passed through.

LONG, BLOB, and columns selected in *Sequencing column* and *Row operation column* options are not allowed in the *Define columns to filter updated rows* list.

> Note
>
> Because the *Define columns to filter updated rows* option filters out CDC updated rows, a target table should have primary key that identifies the row to update the change record.

3.5.3.8 Pivot (Columns to Rows)

Creates a new row for each value in a column that you identify as a pivot column.

The Pivot transform allows you to change how the relationship between rows is displayed. For each value in each pivot column, Data Services produces a row in the output data set. You can create pivot sets to specify more than one pivot column.

3.5.3.8.1 Data inputs

A data set with rows flagged as NORMAL.

3.5.3.8.2 Options

Option	Description
Data field column	The name of the column that contains the pivoted data. This column contains all of the *Pivot columns* values. The data type of this column is determined by the data type of *Pivot columns*. If two or more *Pivot columns* contain different data types, Data Services converts the columns to a single data type—the data type of the first column you add to the pivot set.

Option	Description
Header column	The name of the column that contains the pivoted column names. This column lists the names of the columns where the corresponding data originated.
Non-pivot columns	The columns in the source that are to appear in the target without modification.
Pivot columns	A set of columns to be rotated into rows. Describe these columns in the *Header column*. Describe the data in these columns in the *Data field column*.
Pivot sequence column	The name you assign to the sequence number column. For each row created from a pivot column, Data Services increments and stores a sequence number. Data Services resets the sequence to 1 when creating a row from an original row. For example, if the row corresponds to the first column pivoted, the sequence number for the row is 1.
Pivot set	The number that identifies a pivot set. For each pivot set, you define a group of pivot columns, a pivot data field, and a pivot header name. Each pivot set must have a unique *Data field column* and the *Header column*. Data Services automatically saves this information.

3.5.3.8.3 Data outputs

A data set with rows flagged as NORMAL. This target includes the nonpivoted columns, a column for the sequence number, the data field column, and the pivot header column.

3.5.3.8.4 Example

Suppose you have a table containing rows for an individual's expenses, broken down by expense type.

Emp_name	Mgr_ID	Internal _Expense	Travel _Expense	Misc _Expense
AAA	1234	2000.00	5000.00	100.00
BBB	9876	3000.00	0.00	1000.00
CCC	5555	4800.00	800.00	0.00

This source table has expense numbers in several columns, so you might have difficulty calculating expense summaries. The Pivot transform can rearrange the data into a more manageable form, with all expenses in a single column, without losing category information.

Set the Pivot transform options to pivot the data such that all of the expenses are in the same column. Note that you only need one pivot set in this example.

Options	Value
Pivot sequence column	Sequence
Nonpivot columns	Emp_name
Pivot set	1
Data field column	Expense
Header column	Expense_Type
Pivot columns	Internal_Expense Travel_Expense Misc_Expense

Select *ViewRefresh* or press *F5* to update the output schema for the Pivot transform.

The output data set includes the employee name (not pivoted) and new columns for the pivot sequence, expense type (pivot header), and actual expense data. The manager ID column is not listed in either the pivot or the nonpivot column lists, so it is not included in the output.

The result is a single column of expense values that can be summarized easily.

Emp_name	Sequence	Expense_Type	Expense
AAA	1	Internal_Expense	2000.00
AAA	2	Travel _Expense	5000.00
AAA	3	Misc _Expense	100.00
BBB	1	Internal_Expense	3000.00
BBB	2	Travel _Expense	0.00
BBB	3	Misc _Expense	1000.00
CCC	1	Internal_Expense	4800.00
CCC	2	Travel _Expense	800.00
CCC	3	Misc _Expense	0.00

Suppose that instead of containing one type of data—expenses—your source table contains two types of data—expenses and days traveling both domestically and internationally—for two months.

Emp_name	Dom_Exp	Int_Exp	Dom_Day	Int_Day
AAA	2000.00	5000.00	10	5
BBB	3000.00	0.00	0	0
CCC	4800.00	800.00	15	1

You want to create a target table that has the data in two columns: expenses and days. Therefore, you want to create two pivot sets. Create one pivot set to pivot on the expense columns.

Options	Value
Pivot sequence column	Seq
Nonpivot columns	Emp_name
Pivot set	1
Data field column	Expense
Header column	Expense_Type
Pivot columns	Dom_Exp Int_Exp

Create a second pivot set to pivot on the day columns.

Options	Value
Pivot sequence column	Seq
Nonpivot columns	Emp_name
Pivot set	2
Data field column	Num_Days
Header column	Day_Type
Pivot columns	Dom_Day Int_Day

In this case, the output data set includes the employee name (not pivoted) and new columns for the pivot sequence, expense type, expense data, day type, and day data. Because you linked the pivot data, domestic and international data are contained in unique rows.

Emp_name	Seq	Expense_Type	Expenses	Day_Type	Num_Days
AAA	1	Dom_Exp	2000.00	Dom_Day	10
AAA	2	Int_Exp	5000.00	Int_Day	5
BBB	1	Dom_Exp	3000.00	Dom_Day	12
BBB	2	Int_Exp	0.00	Int_Day	0
CCC	1	Dom_Exp	4800.00	Dom_Day	15

Emp_name	Seq	Expense_Type	Expenses	Day_Type	Num_Days
CCC	2	Int_Exp	800.00	Int_Day	1

When working with multiple pivot sets, pivoted columns cannot contain a different number of rows.

If the example target table contained additional expenses (internal plus miscellaneous expenses), but only had days traveled to match domestic and international travel expenses, the expense data set would be larger than the days traveled data set. In that case, you would have to add a new artificial column containing NULL values to the input data set, and associate the day columns with those additional expenses.

Emp_name	Seq	Expense_Type	Expenses	Day_Type	Num_Days
AAA	1	Dom_Exp	2000.00	Dom_Day	10
AAA	2	Int_Exp	5000.00	Int_Day	5
AAA	3	Internal_Exp	500.00	NULL	NULL
AAA	4	Misc_Exp	75.00	NULL	NULL
BBB	1	Dom_Exp	3000.00	Dom_Day	12
BBB	2	Int_Exp	0.00	Int_Day	0
BBB	3	Internal_Exp	350.00	NULL	NULL
BBB	4	Misc_Exp	140.00	NULL	NULL
CCC	1	Dom_Exp	4800.00	Dom_Day	15
CCC	2	Int_Exp	800.00	Int_Day	1
CCC	3	Internal_Exp	1000.00	NULL	NULL
CCC	4	Misc_Exp	55.00	NULL	NULL

3.5.3.9 Reverse Pivot (Rows to Columns)

Creates one row of data from several existing rows.

The Reverse Pivot transform allows you to combine data from several rows into one row by creating new columns. For each unique value in a pivot axis column and each selected pivot column, Data Services produces a column in the output data set.

3.5.3.9.1 Data inputs

A data set with rows flagged as NORMAL.

3.5.3.9.2 Options

Option	Description
Axis value	The value of the pivot axis column that represents a particular set of output columns. A set of *Pivoted columns* is generated for each axis value. There should be one *Axis value* for each unique value in the *Pivot axis column*.
Column Prefix	Text added to the front of the *Pivoted column* names when creating new column names for the rotated data. An underscore separates the prefix name from the pivoted column name.
Default value	The value stored when a rotated column has no corresponding data. The default is "null" if you do not enter a value. Do not enter a blank.
Duplicate value	Action taken when a collision occurs. A collision occurs when there is more than one row with the same key and value in the *Pivot axis column*. In this case, you can store either the first row or the last row, or you can abort the transform process.
Input data is grouped	Select to indicate whether the input rows are already sorted based on columns specified in the *Non-pivot columns* box. This can improve the performance of the transform.
Non-pivot columns	The columns in the source table that will appear in the target table without modification.
Pivot axis column	The column that determines what new columns are needed in the output table. At run time, a new column is created for each *Pivoted column* and each unique value in this column.
Pivoted columns	The columns containing data you want rotated into the same row. A set of columns will be created for each unique value in the *Pivot axis column*.

3.5.3.9.3 Data outputs

A data set with rows flagged as NORMAL. This target includes the nonpivoted columns and a column for the combination of each pivot column and each pivot axis.

3.5.3.9.4 Example

Suppose you had a table containing contact information for each employee. Each row in the table contains data for a particular employee and contact type.

EmpNo	Type	Name	Address	Phone
100	emergency	Andrew	404 Hallam St	555-4450
100	home	Pat	125 Mercury St	555-6035
100	work	Sean	8400 Page Mill Rd	555-5000
200	emergency	Linda	126 River Rd	555-1087
200	home	David	479 Mill St	555-6914
300	work	Joanne	9500 Page Mill Rd	555-8500

Because the table can have several rows for each employee, finding information, such as a missing contact, for a particular employee may be difficult. The Reverse_Pivot transform can rearrange the data into a more searchable form without losing the category information.

Set the Reverse_Pivot transform options to pivot the data such that all of the contact information for a particular employee is in the same row.

Option	Value		
Non-pivot columns	EmpNo		
Pivoted columns	Name Phone		
Default value	Null		
Pivot axis column	Type		
Duplicate value	Abort		
Axis Value	emergency	home	work
Column Prefix	emergency	home	work

The output data set includes the employee number field (not pivoted) and two fields—name and phone—for each pivot axis. In this case, there are three pivot axes (emergency, home, and work). Therefore, there are six additional fields. In cases where there is no data for a field in the initial source, the Reverse_Pivot transform stores a null value.

The result is a single row for each employee, which you can use to search easily for missing contact information.

EmpNo	Emerg_Name	Emerg_ Phone	Home_Name	Home_ Phone	Work_Name	Work_ Phone
100	Andrew	555-4450	Pat	555-6035	Sean	555-5000
200	Linda	555-1087	David	555-6914	Null	Null
300	Null	Null	Null	Null	Joanne	555-8500

3.5.3.10 Table_Comparison

Compares two data sets and produces the difference between them as a data set with rows flagged as INSERT, UPDATE, or DELETE.

The Table_Comparison transform allows you to detect and forward changes that have occurred since the last time a target was updated.

Note that in order to use the Table_Comparison transform with Teradata 13 and later tables as the comparison table and target table, you must do the following:

- On the Teradata server, set the *General* parameter *DBSControl* to TRUE to allow uncommitted data to be read.
- In the Data Services Teradata datastore, add the following statement in the "Additional session parameters" field:

```
SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
```

Related Information

Designer Guide: Distributed data flow execution [page 301]
Teradata [page 902]

3.5.3.10.1 Data inputs

- The data set from a source or the output from another transform. Only rows flagged as NORMAL are considered by the transform. This data is referred to as the *input data set*.
- The specification for a database table to compare to the input data set. This table is referred to as the *comparison table*.

If the input data set contains hierarchical (nested) data, Data Services includes only the top-level data in the comparison and does not pass nested schemas through to the output.

Use caution when using columns of data type `real` in this transform. Comparison results are unpredictable for this data type.

Related Information

Operation codes [page 1068]

3.5.3.10.2 Options

To use the optional options, consider the type of data that might have changed since you last updated the target warehouse. Have individual records likely been changed more than once (for example, do duplicate primary key rows exist or might they be encountered during job processing)? If so, do you want to record all changes or just the latest information?

If the columns you specify for the *Input primary keys* option have a unique key per row, then you do not need to use the *Input contains duplicate keys* option.

If a record (row) might change more than once during job execution, consider a plan to handle duplicate keys:

- If you assume the input data set has duplicate keys, then to avoid data corruption, select the *Input contains duplicate keys* option. This option allows the Table_Comparison transform to output all duplicate key rows in the input data set columns.
- If you assume that the comparison table has duplicate keys, then to avoid data corruption, use the *Generated Key column* option. This option reads only the largest in a set of duplicated keys.
- If you use the *Generated Key column* option with the *Detect Deleted row(s) from comparison table* option, you can specify whether to detect deletes in all duplicate rows or in the row with the largest generated key.

The following table provides more detail about the Table_Comparison options.

Option	Description
Compare columns	(Optional) Improves performance by comparing only the subset of columns you drag into this box from the input schema. If no columns are listed, all columns in the input data set that are also in the comparison table (that are not of the long or blob data type or the Generated key column) are used as compare columns. You do not need to add primary key columns to the compare column list. They are always compared before the compare columns apply. The compare columns apply only if the primary key value from the input data set matches a value in the comparison table. If the primary key value from the input data set does not match a value in the comparison table, Data Services generates an INSERT row without further comparisons. If the primary key value from the input data set matches a value in the comparison table and values in the non-key compare columns differ in the corresponding rows from the input data set and the comparison table, Data Services generates an UPDATE row with the values from the input data set row.
Comparison method	Select a method for accessing the comparison table: ● *Row-by-row select* — Select this option to have the transform look up the target table using SQL every time it receives an input row. This option is best if the target table is large compared to the number of rows the transform will receive as input. ● *Cached comparison table* — Select this option to load the comparison table into memory. In this case, queries to the comparison table access memory rather than the actual table. This option is best when you are comparing the entire target table. Data

Option	Description
	Services uses pageable cache as the default. If the table fits in the available memory, you can change the cache type to in-memory in the data flow Properties. • *Sorted input* — Select this option to read the comparison table in the order of the primary key column(s) using sequential read. This option improves performance because Data Services reads the comparison table only once. To take advantage of this option, the order of the input data set must match the order of all primary key columns in the Table_Comparison transform. If this is already the case, drag the primary key columns from the input schema in the Table_Comparison transform into the Input primary key columns box. Using a sequential read, Data Services reads the comparison table in the order of the primary key columns. If you must pre-sort the input data, add a query between the source and the Table_Comparison transform. Then, from the query's input schema, drag the primary key columns into the Order By box of the query. The query's columns, in the Order By box, must match the order of the primary key columns that you drag into the Input primary key columns box of the Table_Comparison transform. In this way, you explicitly add a sort operation (ORDER BY) to the query's input data set to ensure that the input data set order will match the order of the read from the comparison table data in the transform.
Detect deleted row(s) from comparison table	(Optional) Generates DELETEs for all rows that are in the comparison table and not in the input set. Assumes the input set represents the complete data set. By default this option is turned off. Note The Table_Comparison transform can flag rows as DELETE. However, rows actually become DELETE rows only after the data flow completes all other processing • An additional section appears in the Table_Comparison editor, which allows you to specify how to handle DELETE rows with duplicate keys, if you select the following options: ○ *Generated key column* ○ *Detect deleted row(s) from comparison table* ○ *Row-by-row select* or the *Sorted input* comparison method If you choose the above options, choose one of the following for deleted rows with the same key value: ○ *Detect all rows*: Output detected DELETEs for all rows.

Option	Description
	○ *Detect row with largest generated key value*: Output detected DELETEs for only the generated key row with the largest value. ● If you select this option and the *Cached comparison table* option, Data Services always deletes the row with largest generated key value. If you do not specify a *Generated key column*, then the sub-options with the *Detect Deleted row(s) from comparison table* option are not enabled. If Data Services finds duplicate keys in the comparison table, then the DELETE output is corrupted. **Note** If you choose the *Detect Deleted row(s) from comparison table* option, then the performance of the data flow will be slower. The comparison method most affected is the *Row-by-row select*, followed by *Cached compare* table, then *Sorted input* option. For *Row-by-row select* and *Cached compare* table, Data Services processes the deleted rows at the end of the data flow. For *Sorted input*, Data Services processes deleted rows as they are encountered in the data flow. *Run as a separate process* Creates a separate sub data flow process for the Table_Comparison transform.
Generated Key Column	(Optional) Provides a method of handling duplicate keys in the comparison table. If for example, you have employee data that includes a social security number as a primary key, and multiple entries for some of these keys, specify a column in the comparison table with unique keys as the generated key column. A generated key column indicates which row of a set containing identical primary keys is to be used in the comparison. Specify an existing column name that by design contains no duplicate keys. The Generated key column option insures that: ● For an UPDATE, the output data set will contain the largest generated key found for a given primary key. For an INSERT, the output data set will contain a NULL value for the generated key, because this column is typically omitted from the input into the Table Comparison transform and set later by a Key_Generation. If an input column is used as generated key column, its value is preserved. For a DELETE, the output data set can include all duplicate key rows or just the row with the largest key value. ● If there is more than one row in the comparison table with the same primary key value and generated key value, the transform arbitrarily chooses which row to compare.

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><ul><li>If the input data set and the comparison table both have the column you specified in <i>Generated key column</i>, the transform does not compare the values for this column; it preserves the value.</li></ul>Without his option:<ul><li>If your comparison table contains rows with the same primary keys, the transform arbitrarily chooses which of these rows to compare.</li></ul></td></tr>
<tr><td><i>Input contains duplicate keys</i></td><td>(Optional) Provides a method of handling duplicate keys in the input data set. If you have more than one row with the same key in the <i>Input Primary Key</i> box, then select this check box. Data Services processes all duplicate rows. Inserts, updates, and deletes occur in the same order as they occur in the input table.

If your input columns have duplicate keys and you do not select this option, then the transform arbitrarily chooses which of these rows to compare during data flow processing.

This option uses additional memory to track rows with duplicate keys. It is recommended that you use this feature only as needed.</td></tr>
<tr><td><i>Input primary key column(s)</i></td><td>The input data set columns that uniquely identify each row. These columns must be present in the comparison table with the same column names and data types.

Drag the column(s) from the input schema into the <i>Input primary key columns</i> box. The transform selects rows from the comparison table that match the values from the primary key columns in the input data set.

If values from more than one column are required to uniquely specify each row in the table, add more than one column to the <i>Input primary key columns</i> box.

You cannot include nested schemas in the <i>Input primary key columns</i> list.</td></tr>
<tr><td><i>Table name</i></td><td>The fully qualified name of the source table from which the maximum existing key is determined (key source table). This table must be already imported into the repository. <i>Table name</i> is represented as "DATASTORE.OWNER.TABLE" where DATASTORE is the name of the datastore Data Services uses to access the key source table; OWNER, if required, depends on the database type associated with the table:
<table>
<tr><th>Database type</th><th>Owner value</th></tr>
<tr><td>DB2</td><td>Data source dependent</td></tr>
<tr><td>Informix</td><td>Informix-defined user name</td></tr>
<tr><td>Microsoft SQL Server</td><td>User name</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><table>
<tr><th>Database type</th><th>Owner value</th></tr>
<tr><td>ODBC</td><td>Data source dependent</td></tr>
<tr><td>Oracle</td><td>User name</td></tr>
<tr><td>SAP Sybase SQL Anywhere</td><td>User Name</td></tr>
<tr><td>SAP Sybase</td><td>User name</td></tr>
</table></td></tr>
<tr><td><i>Filter</i></td><td>(Optional) Limits the rows from the comparison table that are considered for comparison against the input data set.
⚠ Caution
Incorrect filtering can produce false <code>INSERT</code> rows. Construct your filter expression carefully to avoid unexpected results.
Valid filter expressions may contain the following elements:<ul><li>Columns from the comparison table</li><li>Constants (integers, strings, substitution parameters, global variables, and expressions with these combinations)</li><li>Relational operators (<code>=</code>, <code><</code>, <code>></code>, <code><=</code>, <code>>=</code>, <code>LIKE</code>, <code>IN</code>)</li><li>Logical operators (<code>AND</code>, <code>OR</code>)</li></ul>ℹ Note
Data Services functions, and <code>JOIN</code> and other SQL statements can not be used in filter expressions.
Examples of valid filter expressions:<ul><li><code>TC.col1 IN (1, 2, 3)</code></li><li><code>TC.col1 >= 1 AND TC.col1 < 1000</code></li><li><code>TC.col1 = 1 AND TC.col2 = 2</code></li><li><code>col1 = 10 OR col1 = 20</code></li><li><code>TC.col1 = $V_DEPTNO</code></li></ul></td></tr>
</table>

3.5.3.10.3 Data outputs

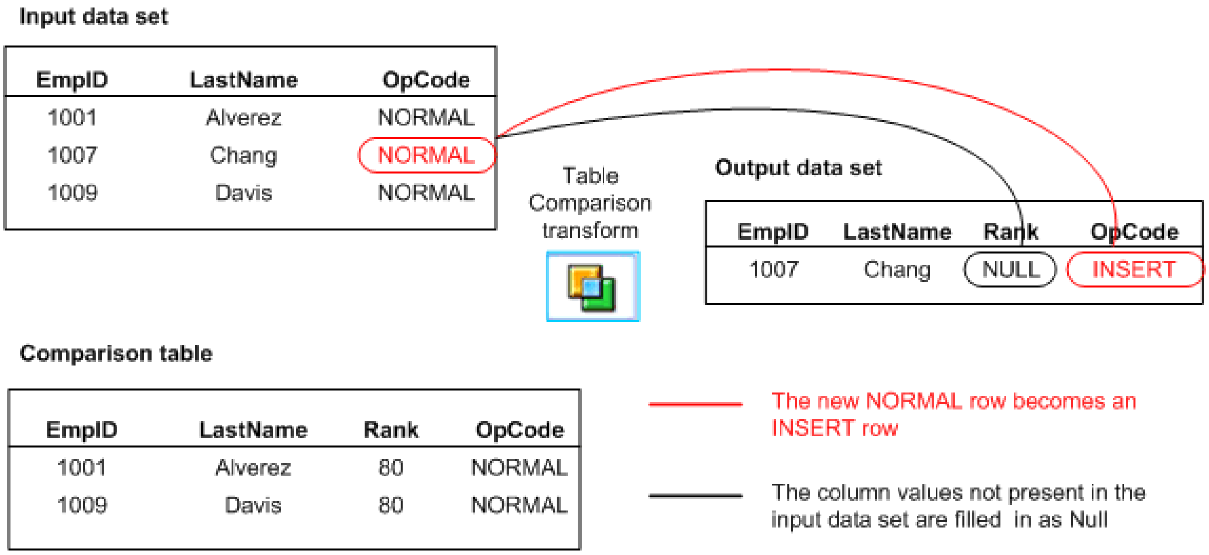
A data set containing rows flagged as INSERT, UPDATE, or DELETE. This data set contains only the rows that make up the difference between the two input sources: one from the input to the transform (input data set), and one from a database table you specify in the transform (the comparison table). The transform selects rows from the comparison table based on the primary key values from the input data set. The transform compares columns that exist in the schemas for both inputs.

If a column has a `date` data type in one table and a `datetime` data type in the other, the transform compares only the date section of the data. The columns can also be `time` and `datetime` data types, in which case Data Services only compares the time section of the data.

The transform generates a data set consisting of rows with INSERT and UPDATE operation codes, unless you are using the *Detect Deleted row(s) from comparison table* option. In this case, DELETE rows are produced. If a primary key value in the comparison table is not present in the input data set, no corresponding row appears in the output.

For each row in the input data set, there are four possible outcomes from the transform:

- An INSERT row.
  The primary key value from the input data set does not match a value in the comparison table. The transform produces an INSERT row with the values from the input data set row.
  If there are columns in the comparison table that are not present in the input data set, the transform adds these columns to the output schema and fills them with NULL values.



Input data set

EmpID	LastName	OpCode
1001	Alverez	NORMAL
1007	Chang	NORMAL
1009	Davis	NORMAL

Output data set

EmpID	LastName	Rank	OpCode
1007	Chang	NULL	INSERT

Comparison table

EmpID	LastName	Rank	OpCode
1001	Alverez	80	NORMAL
1009	Davis	80	NORMAL

Table
Comparison
transform
The new NORMAL row becomes an INSERT row
The column values not present in the input data set are filled in as Null

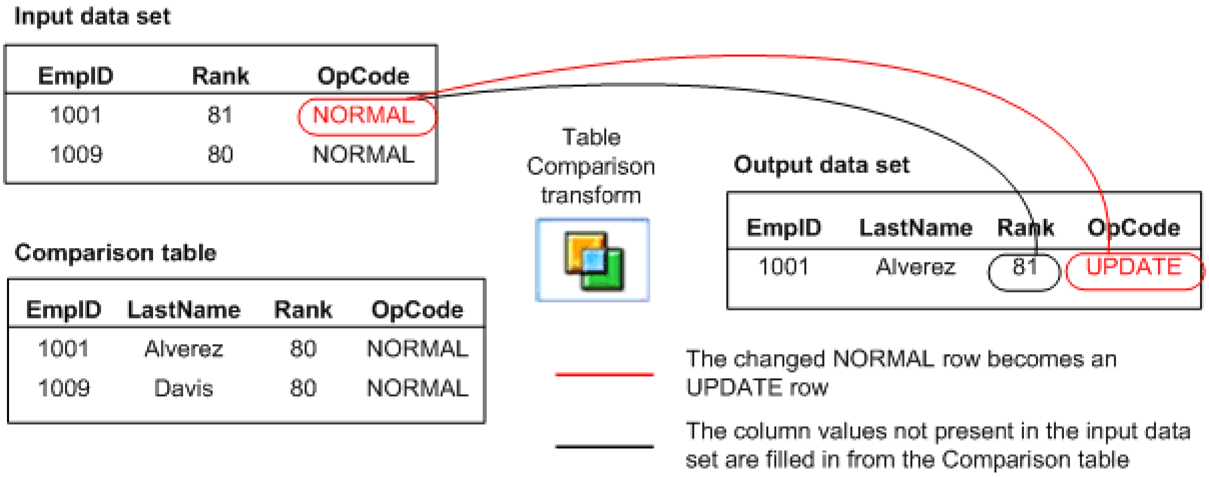
> **Note**
>
> Data Services ignores trailing blanks when it compares values in the comparison table and the input data set. However, an Oracle database server includes trailing blanks in comparisons. Therefore, the Table_Comparison transform produces an INSERT row under the following circumstances:

  - The comparison table is an Oracle table with data that had trailing blanks removed.
  - The input data contains trailing blanks.
  - You use the *Row-by-row select* comparison method. This method pushes down the comparison to the Oracle database server.

  To avoid inserting rows when the data differs only by number of trailing blanks, take either of the following actions:

  - Use a different comparison method (*Cached comparison table* or *Sorted input*) if possible
  - Add the rtrim or rtrim_blank function to remove trailing blanks from the input data.
- An UPDATE row.

The primary key value from the input data set matches a value in the comparison table, and values in the non-key compare columns differ in the corresponding rows from the input data set and the comparison table. The transform produces an UPDATE row with the values from the input data set row.
If there are columns in the comparison table that are not present in the input data set, the transform adds these columns to the output schema and fills them with values from the comparison table.



Input data set

EmpID	Rank	OpCode
1001	81	NORMAL
1009	80	NORMAL

Comparison table

EmpID	LastName	Rank	OpCode
1001	Alverez	80	NORMAL
1009	Davis	80	NORMAL

Output data set

EmpID	LastName	Rank	OpCode
1001	Alverez	81	UPDATE

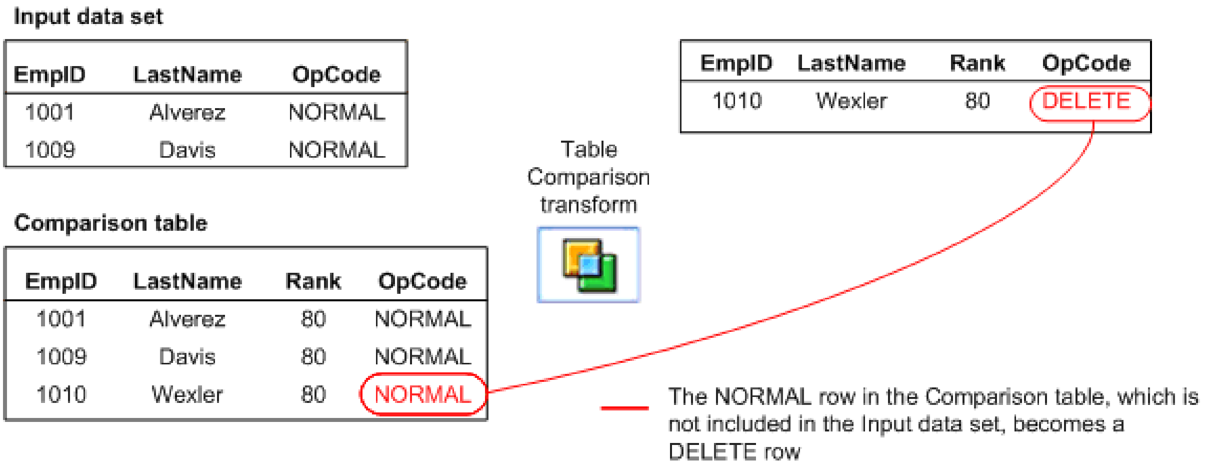
🟥 The changed NORMAL row becomes an UPDATE row

― The column values not present in the input data set are filled in from the Comparison table

> **Note**
>
> Data Services ignores trailing blanks when it compares values in the comparison table and the input data set. Therefore, if the input data and the value in the comparison table differ only by trailing blanks, the transform will not produce an UPDATE row. However, an Oracle database server includes trailing blanks in comparisons. To avoid updating rows when the data differs only by number of trailing blanks, take either of the following actions:

- Use a different comparison method (*Cached comparison table* or *Sorted input*), if possible.
- Add the rtrim or rtrim_blank function to remove trailing blanks from the input data.

- A DELETE row.
  The primary key value from the comparison table set does not match a value in the input data set. The transform produces an DELETE row with the values from the comparison table row.
  If there are columns in the comparison table that are not present in the input data set, the transform adds these columns to the output schema and fills them with values from the comparison table.



Input data set

EmpID	LastName	OpCode
1001	Alverez	NORMAL
1009	Davis	NORMAL

Comparison table

EmpID	LastName	Rank	OpCode
1001	Alverez	80	NORMAL
1009	Davis	80	NORMAL
1010	Wexler	80	NORMAL

EmpID	LastName	Rank	OpCode
1010	Wexler	80	DELETE

The NORMAL row in the Comparison table, which is not included in the Input data set, becomes a DELETE row

- The row is ignored.
  The primary key value from the input data set matches a value in the comparison table, but the comparison does not indicate any changes to the row values.

Related Information

Operation codes [page 1068]

3.5.3.11 XML_Pipeline

Processes large XML files one instance of a repeatable structure at a time.

With this transform, Data Services does not need to read the entire XML input into memory then build an internal data structure before performing the transformation. An NRDM structure is not required to represent the entire XML data input. Instead, the XML_Pipeline transform uses a portion of memory to process each instance of a repeatable structure, then continually releases and reuses memory to steadily flow XML data through the transform.

During execution, Data Services pushes operations of the XML_Pipeline transform to the XML source.

3.5.3.11.1 Data inputs

XML file or XML message.

You can connect more than one XML_Pipeline transform to an XML source.

3.5.3.11.2 Editor

Use the XML_Pipeline transform editor to specify:

- Input schema
- Output schema

The XML_Pipeline transform editor was streamlined to support massive throughput of XML data, therefore it does not contain additional options. In addition to input and output schemas, the Mapping tab shows how Data Services will map any selected output column.

When connected to an XML source, the XML_Pipeline transform editor shows the input and output schema structures as a root schema containing repeating and non-repeating sub-schemas represented by the following icons:

▦	Root schema and repeating sub-schema
▤	Non-repeating sub-schema

Within each sub-schema, mapped and unmapped columns display as follows:

▯	Column not used in output mapping
▶	Column used in output mappings

3.5.3.11.3 Rules

Rules

- You cannot drag and drop the root level schema.
- You can drag and drop the same child object repeated times to the output schema only if you give each instance of that object a unique name. You must rename the mapped instance before attempting to drag and drop the same object to the output again.
- When you drag and drop a column or sub-schema to the output schema, you cannot then map the parent schema for that column or sub-schema. Similarly, when you drag and drop a parent schema, you cannot then map an individual column or sub-schema from under that parent.
- You cannot map items from two sibling repeating sub-schemas because the XML_Pipeline transform does not support Cartesian products (combining every row from one table with every row in another table) of two repeatable schemas.

3.5.3.11.4 Limitations

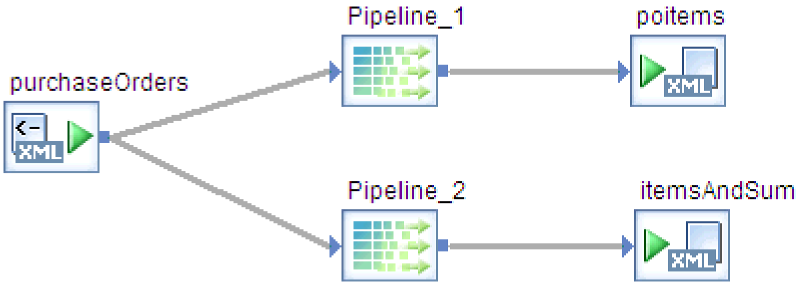
Limitations

The XML_Pipeline transform:

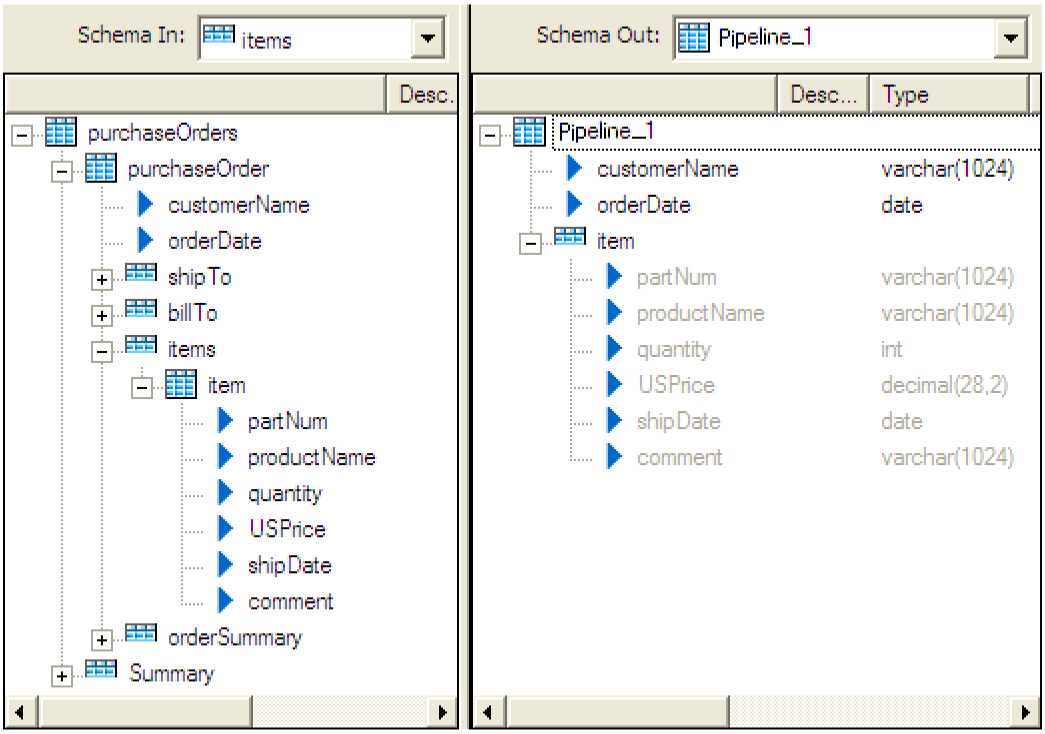
- Pushes operations to the XML source during transform execution, so you cannot use a breakpoint between your XML source and a XML_Pipeline transform.

3.5.3.11.5 Example

This simple data flow contains two XML_Pipeline transforms, but only one of them takes full advantage of the pipelining power.



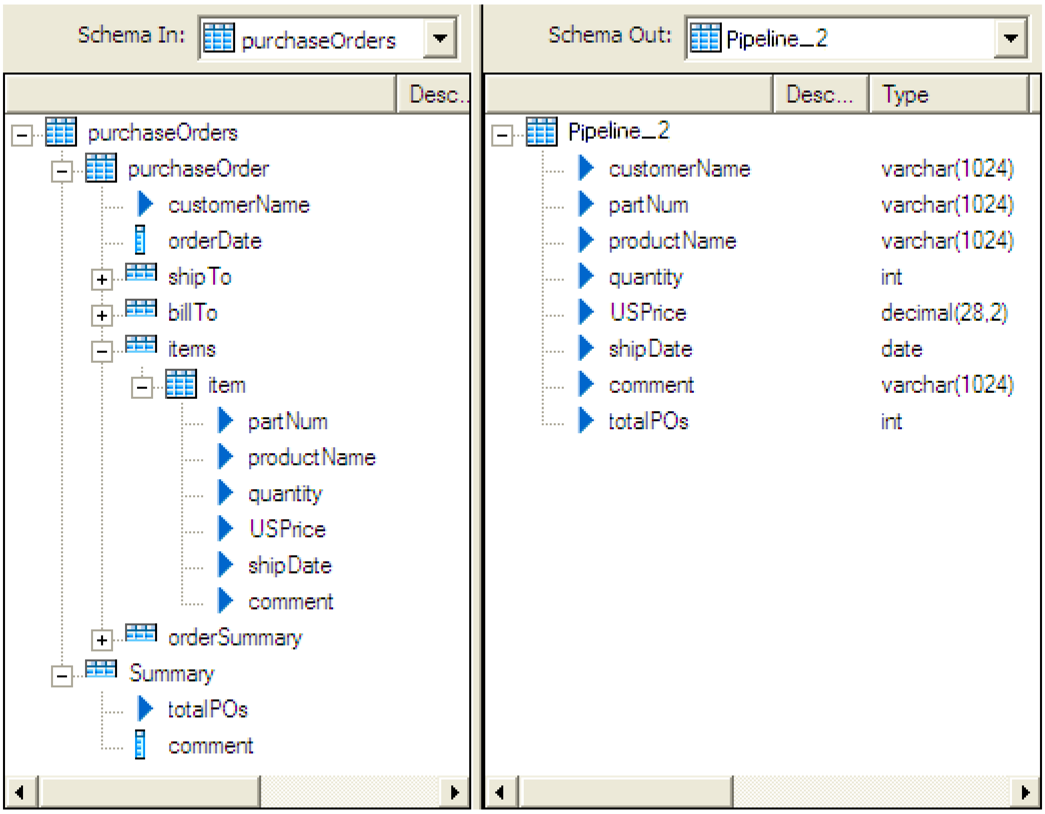
- The "XML_Pipeline_1" transform allows XML data to flow because the repeatable column, purchaseOrders.purchaseOrder.items.item is selected. The XML source produces one row after parsing one item.



Schema In: items

Schema Out: Pipeline_1

- The "XML_Pipeline_2" transform does not take advantage of the pipelining power because the purchaseOrders.Summary.totalPOs column is selected and this column occurs structurally after the repeatable column (purchaseOrders.purchaseOrder.items.item). In this scenario, the XML source must assemble the entire structure of items in memory before processing.



Pipeline_2	Type
customerName	varchar(1024)
partNum	varchar(1024)
productName	varchar(1024)
quantity	int
USPrice	decimal(28,2)
shipDate	date
comment	varchar(1024)
totalPOs	int

However, if you broke this up into two XML_Pipeline transforms, mapping the purchaseOrders.Summary.totalPOs column in a separate transform, you could connect both XML_Pipeline transforms to a Query transform and take advantage of the reduced memory consumption to get the same output.

Related Information

Query [page 1441]

3.5.4 Data Quality transforms

3.5.4.1 Transform configurations

The following Data Quality transform configurations are available from the *Transforms* tab of the object library. This section describes these transform configurations in detail.

Table 147: Associate transform

Transform configuration	Description
AssociateGroupStatistics_AssociateBatch	A sample Associate transform configured with group statistics.

Transform configuration	Description
Base_Associate	A base Associate transform that is used to configure an Associate transform.
Wizard_AssociateBatch	A sample Associate configuration used by the Match Wizard. This configuration should not be edited.

Table 148: Country ID transform

Transform configuration	Description
CountryID2Char	A sample Country ID transform configured to generate the two-character ISO country code.

Table 149: Data Cleanse transform

Transform configuration	Description
Base_DataCleanse	A base Data Cleanse transform that is used to configure a custom Data Cleanse transform.
Chinese_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, email, and phone data using Chinese-based data quality rules.
Dutch_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, email, and phone data using Dutch-based data quality rules.
EnglishNorthAmerica_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, email, and phone data using English-based data quality rules.
French_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, email, and phone data using French-based data quality rules.
German_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, email, and phone data using German-based data quality rules.
Italian_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, email, and phone data using Italian-based data quality rules.
Japanese_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, email, and phone data using Japanese-based data quality rules.
Portuguese_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, email, and phone data using Portuguese-based data quality rules.
Spanish_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, email, and phone data using Spanish-based data quality rules.

Table 150: Geocoder transform

Transform configuration	Description
Geocode	A sample Geocoder transform configured to assign latitude/longitude based on an address or point-of-interest reference point.
ResultListGeocode	A sample Geocoder transform configured to provide a list of addresses or points of interest based on an address or latitude/longitude reference point.
ReverseGeocode	A sample Geocoder transform configured to assign an address based on a latitude/longitude reference point.

Table 151: Global Address Cleanse transform

Transform configuration	Description
Australia_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Australia.
Brazil_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Brazil.
Canada_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Canada.
China_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in China.
Europe_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in multiple European countries.
France_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in France.
Germany_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Germany.
Global_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse Latin-1 address data in any supported country.
GlobalSuggestions_Address_Cleanse	A sample Global Address Cleanse transform configured to cleanse Latin-1 address data in any supported country using the Suggestion List feature.
Greece_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Greece when the address data consists of Greek data.
Italy_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Italy.
Japan_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Japan when the address data consists of Japanese kanji, katakana, and hiragana.
Portugal_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Portugal.

Transform configuration	Description
Spain_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Spain.
UK_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in the United Kingdom.
USA_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in the United States.
USASuggestions_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in the United States using the Suggestion List feature.

Table 152: Global Suggestion List transform

Transform configuration	Description
GlobalSuggestions	A sample Global Suggestion List transform configured to generate a suggestion list for Latin-1 address data in any supported country.
UKSuggestions	A sample Global Suggestion List transform configured to generate a suggestion list for partial address data in the United Kingdom.

Table 153: Match transform

Transform configuration	Description
Address_MatchBatch	A sample Match transform configured to identify matching data records based on similar address data.
AddressJapan_MatchBatch	A sample Match transform configured to identify matching data records based on similar address data when the data consists of Japanese kanji, katakana, and hiragana.
AddressSingleField_MatchBatch	A sample Match transform configured to identify matching data records based on similar address data (when the address data is in a single field).
Base_Match	A base Match configuration that is used to configure a Match transform without necessarily performing matching.
ConsumerHouseholdResFamInd_ MatchBatch	A sample Match transform configured to identify three levels of matching data records: residence based on similar address data, family based on similar family name data, and individual based on similar given name and postname data.
ConsumerHouseholdResInd_MatchBatch	A sample Match transform configured to identify two levels of matching data records: residence based on similar address data, and individual based on similar name data.
CorporateHouseholdFirmInd_MatchBatch	A sample Match transform configured to identify two levels of matching data records: firm based on similar firm and address data, and individual based on similar name data.
FirmAddress_MatchBatch	A sample Match transform configured to identify matching data records based on similar firm and address data.

Transform configuration	Description
FirmAddressJapan_MatchBatch	A sample Match transform configured to identify matching data records based on similar firm and address data when the data consists of Japanese kanji, katakana, and hiragana.
IndividualId_MatchBatch	A sample Match transform configured to identify matching data records based on the same individual identification number data.
NameAddress_MatchBatch	A sample Match transform configured to identify matching data records based on similar name and address data.
NameDate_MatchBatch	A sample Match transform configured to identify matching data records based on similar name and date data.
NameEmail_MatchBatch	A sample Match transform configured to identify matching data records based on similar name and email data.
NameFirmAddress_MatchBatch	A sample Match transform configured to identify matching data records based on similar name, firm, and address data.
NameFirmAddressJapan_MatchBatch	A sample Match transform configured to identify matching data records based on similar name, firm, and address data when the data consists of Japanese kanji, katakana, and hiragana.
NameIndividualId_MatchBatch	A sample Match transform configured to identify matching data records based on similar name and individual identification number data.
NamePhone_MatchBatch	A sample Match transform configured to identify matching data records based on similar name and phone data.
ProductDescription_MatchBatch	A sample Match transform configured to identify matching data records based on similar product description data.
Wizard_MatchBatch	A sample Match configuration used by the Match Wizard. This configuration should not be edited.

Table 154: USA Regulatory Address Cleanse transform

Transform configuration	Description
USARegulatory_AddressCleanse	A sample USA Regulatory Address Cleanse transform configured to cleanse address data according to CASS requirements.
USARegulatoryEWS_AddressCleanse	A sample USA Regulatory Address Cleanse transform configured to cleanse address data according to CASS requirements, with Early Warning System.
USARegulatoryGeo_AddressCleanse	A sample USA Regulatory Address Cleanse transform configured to cleanse address data according to CASS requirements, with GeoCensus.
USARegulatoryNCOALink_AddressCleanse	A sample USA Regulatory Address Cleanse transform configured to cleanse address data using NCOALink data.
USARegulatoryNonCertified_ AddressCleanse	A sample USA Regulatory Address Cleanse transform configured to cleanse address data in non-certified mode.

Transform configuration	Description
USARegulatoryNonCertifiedGeo_ AddressCleanse	A sample USA Regulatory Address Cleanse transform configured to cleanse address data in non-certified mode, with GeoCensus.
USARegulatoryRDI_AddressCleanse	A sample USA Regulatory Address Cleanse transform configured to cleanse address data according to CASS requirements, with Residential Delivery Indicator.
USARegulatorySuggestions_ AddressCleanse	A sample USA Regulatory Address Cleanse transform configured to cleanse address data using the Suggestion List feature.
USARegulatoryZ4Change_ AddressCleanse	A sample USA Regulatory Address Cleanse transform configured to cleanse address data according to CASS requirements, with Z4Change.
DSF2_Walk_Sequencer	A sample DSF2 Walk Sequencer transform configured to append sequence information to your data so that, by using presort software, you can obtain walk sequence discounts.

Table 155: User-Defined transform

Transform configuration	Description
Base_UserDefined	The base User-Defined configuration that is used to configure a User-Defined transform.

Related Information

Designer Guide: Data Flows, To create a transform configuration [page 308]

3.5.4.2 Downloading blueprints and other content objects

We have identified a number of common scenarios that you are likely to handle with SAP Data Services. Instead of creating your own job from scratch, look through the blueprints. If you find one that is closely related to your particular business problem, you can simply use the blueprint and tweak the settings in the transforms for your specific needs.

For each scenario, we have included a blueprint that is already set up to solve the business problem in that scenario. Each blueprint contains the necessary project, jobs, data flows, file formats, sample data, template tables, and custom functions to run the data flows in your environment with only a few modifications.

You can download all of the blueprints or only the blueprints and other content that you find useful from the SAP Community Network website. Here, we periodically post new and updated blueprints, custom functions, best practices, white papers, and other SAP Data Services content. You can refer to this site frequently for updated content and use the forums to provide us with any questions or requests you may have. We have also provided the ability for you to upload and share any content that you have developed with the rest of the development community.

Instructions for downloading and installing the content objects are also located on the Blueprints web page.

1. To access the SAP Data Services Blueprints web page, go to *http://scn.sap.com/docs/DOC-8820* in your web browser.
2. Open the Content Objects User's Guide to view a list of all of the available blueprints and content objects and their descriptions, and instructions for downloading and setting up the blueprints.
3. Select the blueprint that you want to download.
4. Follow the instructions in the user's guide to download the files to the appropriate location and make the necessary modifications in SAP Data Services to run the blueprints.

3.5.4.3 Dynamic transform settings

Dynamic transform settings allow the user to change a transform's settings after the transform is initialized, without having to terminate and reinitialize the transform. You can pass each new setting through an input field to the transform. The transform will get an updated setting from the input field and adjust its processing to use the new setting, before processing the incoming record.

The settings a transform is initialized with are considered the transform's default settings. Dynamic setting values that are specified in the input fields are only valid for that record and do not affect any subsequent record. If the value specified for a given option is NULL or blank, then the record will be processed with the default setting for that option. If the dynamic setting is invalid, then the transform will log a warning and then use the default settings.

The dynamic input fields in Data Services are:

Transform	Dynamic input field
Data Cleanse	Option_Content_Domain_Sequence The domain must be specified as an abbreviation of the domain. The valid predefined values are: AR, ZH, CS, DA, NL, EN_US, EN_GB, EN_AU, EN_IN, FR, DE, HU, ID, IT, JA, MS, NO, PL, PT_BR, PT_PT, RO, RU, SK, ES_MX, ES_ES, SV, and GLOBAL. The Global domain is a special content domain which contains all variations and their associated properties. If a variation is not associated with domain-specific information the Global domain serves as the default domain.The Global domain is required for every content domain sequence. Be sure to add GLOBAL as the last domain in the sequence. The Content domain sequence input field may hold more than one domain. If there is more than one domain, you must separate the domains with a pipe (|). For example, to specify the domain for Spain and then for Portugal, enter **`ES_ES|PT_PT|GLOBAL`**.
Data Cleanse	Option_Output_Format The output format must be specified as an abbreviation of the domain. The valid predefined values are: AR, ZH, CS, DA, NL, EN_US, EN_GB, EN_AU, EN_IN, FR, DE, HU, ID, IT, JA, MS, NO, PL, PT_BR, PT_PT, RO, RU, SK, ES_MX, ES_ES, and SV.
Global Address Cleanse	Option_Canada_Output_Address_Language
Global Address Cleanse	Option_GAC_Dual_Address

Transform	Dynamic input field
Global Address Cleanse	Option_Standardization_Address_Line_Alias
Global Address Cleanse	Option_Standardization_Assign_Locality
Global Address Cleanse	Option_Standardization_Capitalization
Global Address Cleanse	Option_Standardization_Character_Width_Style
Global Address Cleanse	Option_Standardization_Directional_Style
Global Address Cleanse	Option_Standardization_Include_Locality_Addition
Global Address Cleanse	Option_Standardization_Locality_Name_Style
Global Address Cleanse	Option_Standardization_Output_Country_Language
Global Address Cleanse	Option_Standardization_Postal_Phrase_Style
Global Address Cleanse	Option_Standardization_Primary_Type_Style
Global Address Cleanse	Option_Standardization_Region_Style
Global Address Cleanse	Option_Standardization_Secondary_Description_Style
Global Address Cleanse	Option_Standardization_Secondary_Number_Style
Geocoder	Option_Distance_Unit
Geocoder	Option_Max_Records
Geocoder	Option_Radius
Match	Option_Field_Algorithm_Geo_Proximity_**`<logical_name>`**_Max_Distance

3.5.4.4 About Data Quality fields

Many transforms require mapped input and generated output fields. These fields are on the *Input* and *Output* tabs in the transform's editor. The available fields are documented in this section with each transform.

3.5.4.4.1 Content types

The content type identifies the type of data in the field. Setting the content type helps you map your fields when you set downstream transforms. The program searches all upstream fields and automatically maps the fields that have a content type that is relevant to the type of transform that you're currently mapping. Those upstream fields will automatically be added as mapped fields.

For example, in your data source, let's say that you have a column of data that is comprised mostly of first names called Given Name1. This field is automatically mapped to a Data Quality-recognized content type, Given_Name1. You can change the content type in the Output schema. If you are importing an XSD and DTD, you must select *Automatically Assign Content Type* to have the content type automatically assign.

You can change the content type by double-clicking on the field in the Schema Out portion of the transform, or by clicking the Content Type column on the Schema Out grid.

You can specify the content type in your source data including XML Schemas, COBOL Copybooks, flat files, Excel, and IDOC objects. If the data in a column cannot be mapped, then you will see a blank column. Content types in an XML Schema can be changed in the Local Object Library, and the change will be reflected in all data flows where the schema is used. You cannot change the content type in an XML Schema from within a single data flow.

When using some transforms, such as the Query transform, you can change the content type of an input column to a different output content type. You may want to change the content type when you will use the next transform for a different content type, for example, if you have an input column that contains city information with a content type of Locality. When you set up the Query transform, it will return state information to which you would assign the Region content type.

If you attempt to merge the contents of two corresponding columns in the Merge transform, be certain that the content types match. Otherwise, you will see a warning message when you validate.

If you have an input source on your local repository and overwrite it with one from the central repository, then your content type information will be overwritten.

If you re-import a column or table via Designer, your content types for all of the existing columns will be preserved by default. You can change this option to clear all of the content types by choosing ▶ *Tools* › *Options* ◀. Then select ▶ *Designer* › *Attribute Values* ◀. Change *Content Type* from *Preserve* to *Clear* and then the content type will be overwritten by the source data.

Available content types

The following values are available for each transform.

- <Blank>
- Address
- Address Primary Name
- Address Primary Number
- Address Primary Postfix
- Address Primary Prefix
- Address Primary Type
- Address Secondary Number

- Country
- Date
- Delivery Point
- DPV Status
- Email
- Family Name1
- Family Name1 Match Std
- Family Name2
- Family Name2 Match Std
- Firm
- Firm Location
- Firm Location Match Std
- Firm Match Std
- Given Name1
- Given Name1 Match Std
- Given Name2
- Given Name2 Match Std
- Group Number
- Locality
- Lot
- Lot Order
- Name
- Name And Firms
- Phone
- Postcode
- Postcode1
- Postcode2
- Postname
- Postname Match Std
- Prename
- Prename Match Std
- Region
- Sortcode Rte
- SSN
- Title
- Title Match Std

3.5.4.5 Associate

Description

The Associate transform works downstream from your Match transforms to provide a way to combine, or associate, their match results by using the Match transform-generated Group Number fields.

You may want to add a Group Statistics operation to the Associate transform to gather match statistics. You can combine the results of two or more Match transforms, two or more Associate transforms, or any combination of the two.

For example, you may use one Match transform to match on name and address, use a second Match transform to match on SSN, and then use an Associate transform to combine the match groups produced by the two Match transforms.

Related Information

Associate transform options [page 1134]
Association options [page 1134]
Designer Guide: Match, Association matching [page 579]

3.5.4.5.1 Content objects

Transform configurations

A transform configuration is a transform with preconfigured input fields, output fields, and options that can be used in multiple data flows. These are useful if you repeatedly use a transform with specific options and input and output fields.

When Data Services is installed, read-only transform configurations are provided for the Data Quality transforms.

You can use transform configurations in your data flows or as an example of a typical transform. After you place an instance of the transform configuration in a data flow, you can override these preset defaults. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one.

Sample blueprints and other objects

We have created Data Quality blueprints and other content objects to help you set up Data Services jobs. We’ve identified a number of common scenarios that you are likely to perform with Data Services. For each scenario, we’ve included a blueprint that is already set up to solve the business problem in that scenario.

Related Information

Downloading blueprints and other content objects [page 1128]
Transform configurations [page 1123]

3.5.4.5.2 Associate transform options

The following options control transform-level functions.

Option	Description
Associate set name	Specifies the name for your Associate set.
Generate report data	Specifies whether to generate report data for this transform. This option is available in every transform that generates report data. *Yes:* Generates report data for this transform. *No:* Turns off report data generation. If you do not need to generate reports (in production, for example), you should set this option to No. This will improve performance.
Logical source field	Specifies the field that contains the ID for the logical source.
Physical source field	Specifies the field that contains the ID for the physical source (Reader).
Run as a separate process	Specifies whether to split transform into a sub-data flow that can improve performance and throughput by using separate memory and computer resources. *Yes:* Splits transform into sub-data flow that runs on a separate process. *No:* Keeps transform in same process as the rest of the data flow.

Related Information

Performance Optimization Guide: Distributing Data Flow Execution, Run as a separate process [page 2153]

3.5.4.5.3 Association options

This is a repeating option group: you can add as many associations as you want.

Option	Description
Association name	Specifies the name for this association. Be sure that it is unique among other associations in this transform.
Group Number field	Specifies the field that contains the group number data from the Match transform. Add as many of these as you want by clicking the *Add Row* button.

3.5.4.5.3.1 Post-association processing

Use the Post-association processing table to navigate to your operations by double-clicking the desired row in the table.

Best record

The purpose of best record post-processing is to salvage data from matching records—that is, members of match groups—and consolidate, or post, that data to a best record, or to all matching records.

Group statistics

Use group statistics to generate statistical information about your group of matching records. Find out:

- The number of records within the match group
- The sequential group order number
- The group rank, which flags one record within each group of matching records as the Master record and all other records in the group as Subordinate records.
- whether the records in a match group belong to more than one source

Group statistics are essential for generating data for match reports.

Prioritization

Use the prioritization operation to order records for processing by other post-match operations.

Unique ID

You can use the Unique ID options to assign sequential identification numbers to each new record when adding records to a data warehouse. For example, the largest number assigned in a particular project can be carried over as the beginning identification number (plus 1) to be used in the assignment of new sequential IDs. This occurs when the software processes the next source against the data warehouse file.

Output record

Use the Output record options to flag certain types of records for potential processing downstream.

Related Information

Best record options: Best Record tab [page 1136]
Group statistics options [page 1139]
Group prioritization options: Priority Order tab [page 1140]
Unique ID options: Unique ID tab [page 1142]
Output flag selection options [page 1306]
Designer Guide: Match, Best record [page 562]
Designer Guide: Match, Unique ID [page 570]

3.5.4.5.3.1.1 Best record options: Best Record tab

Use the best record post-match processing operation to update your records with information from other records in a match group, among other things.

Option	Description
Best record name	Enter a name for your Best Record operation. Make sure that this name is unique within this Match transform.
Best record strategy	Choose the strategy to determine whether any action is taken on records in a match group. This is the criteria for further action. After you choose the strategy, priority, and field that you want to work with, the Match transform automatically generates the Python code for you (except for Custom). *Custom:* Choose this strategy to base your strategy entirely on custom Python code. This allows you to open the Python Expression editor and create custom Python code. *Date:* Choose Date to base your strategy on a date field. *Length:* Choose Length to base your strategy on the length of data in a field. *Non_Blank:* Choose Non_Blank to base your strategy on the completeness of data in a field. *Priority_Number:* Choose Priority_Number to base your strategy on a number.

Option	Description
	Priority_String: Choose Priority_String to base your strategy on string data in a field.
Strategy priority	These are the choices for priorities for each of the best record strategies, other than Non_Blank and Custom. Date • *Newest:* The newest date in the field will cause an action to take place. • *Oldest:* The oldest date in a field will cause an action to take place. Length • *Longest:* The longest string in a field will cause an action to take place. • *Shortest:* The shortest string in a field will cause an action to take place. Priority_Number • *Highest:* The highest number in a field will cause an action to take place. • *Lowest:* The lowest number in a field will cause an action to take place. Priority_String • *Ascending:* The string with the most ascending string order will cause an action to take place. • *Descending:* The string with the most descending string order will cause an action to take place.
Strategy field	Choose a field that contains data that you need to execute your strategy.
Posting destination	Specifies the destination record. *Master:* Post only to a master record. *Subs:* Post only to subordinate records. *Master to Subs:* Push information from the master record and post it to each subordinate record. *All:* Post to both the master and subordinate records.
Post only once per destination	*Yes:* Post only once per destination record. After data is posted to the destination record, the operation stops. *No:* Post more than once per destination. After data is posted to the destination record, the operation continues and the destination record is populated again with the next value. This option should be used when accumulating values such as total sales. Set this option to Yes when you are using the NON_BLANK strategy. Set this option to No when you are using the Longest, Shortest, Newest, Oldest, Ascending, or Descending priorities.

Option	Description
	ℹ Note This option is ignored when using the *Master to Subs* posting destination. With this posting destination, information can be posted only once.
View/Edit Python	The View/Edit Python button opens the Python Expression editor. If you chose the Custom strategy, you can create your custom Python code. If you chose any other strategy, Python viewed in the editor is read-only.

Best record action fields

Use the best record action fields table to define the actions taken on the fields based on your strategy.

Option/Option group	Description
Source field	Specifies the name of the source field in the input record.
Destination field	Specifies the name of the destination field in the output record, or the destination of this best record action. You can have the action post to the same input field, or you can post to a different field.
Custom	*Yes:* Specifies that you want to create custom Python code to perform an action on the destination field. *No:* Specifies that you want to use the same source and destination fields. When this option is set to No, the contents of the source field are copied to the destination field.
Editor	If you chose Yes in the Custom column, a button appears here to allow you open the Python Expression editor and configure your Python code. You can open the Python Expression editor only if Custom is set to Yes.

Related Information

Designer Guide: Match, Best record [page 562]

3.5.4.5.3.1.2 Best record options: Destination Protection tab

Protect data from being changed by enabling and defining destination protection.

Option	Description
Best record name	Enter a name for your Best Record operation. Make sure that this name is unique within this Match transform.
Enable destination protection	Select to protect records from best record operations that may modify the contents.
Default destination protection	Select the default destination protection setting. This is useful because the default setting will account for records that are protected (or not protected) through the use of sources or fields.
Specify destination protection by field	Select to enable destination protection through a value in a field.
Destination protection field	Choose the field that holds the destination protection value. The field must contain a value of Y or N. Any other value (including blank) will cause the default destination protection specification to occur, if you specified a default destination protection.
Specify destination protection by source	Select this option to control destination protection through membership in a particular source. Fill in the table with source names and whether they are protected.
Source Name	Choose the name of the source from the drop-down list. The list here is populated with defined sources and source groups from the *Input Sources Editor* window of the Match Editor.
Destination protected	Select a value to assign to the source. Select Yes to enable destination for that source. Select No, if you do not wish to protect records from that source.

3.5.4.5.3.1.3 Group statistics options

The Group Statistics option group includes the following options:

Option	Description
Group statistics name	Choose a name for this group statistics operation. If you are including more than one group statistics operation in this Match transform, make sure that the name is unique.
Generate only basic statistics	Select if you want to generate match group statistics. These will not include any statistics about input sources.
Generate source statistics from input sources	Select to generate statistic counts about your input sources. You must have input sources defined in the Match editor for this option to be active. If you do not check this, the Match transform will still generate statistics about your match groups.
Generate source statistics from source values	Select to generate source statistics based on source values in a field. If you have a source value field, using this option, you can choose to count all sources or specific ones based on a particular value.

Option	Description
Logical source field	Specifies the field that holds the value for your logical sources.
Default logical source value	Specifies a value to use if the field in the Logical source field option is blank. For example, if a record has a blank value in the field, this default value is used.
Count all sources	Select to count all sources, no matter what the value in the Logical source field is.
Choose sources to count	Select to specify particular sources to count, based on values in the Logical source field.
Default count flag value	Specifies the value to use when the Predefined count flag field is invalid (for example, if the field has data other than Y or N) or it is empty. *Yes:* Counts all of the records in the source. *No:* Does not count any of the records in the source.
Auto generate sources	Select to generate sources based on the value in a field.
Predefined count flag field	Specifies the field name that contains the indicator value (Y or N) to determine whether a source is counted.

Manually define logical source count flags

Be sure to fill in both columns for this to work.

Option	Description
Source value	Specifies the value in the field to find. This value is case sensitive.
Count	Specifies whether you want to use the value you entered in the Logical source value option in the count. *Yes:* Includes the logical source value in the count. *No:* Does not include the value in the logical source value option in the count.

3.5.4.5.3.1.4 Group prioritization options: Priority Order tab

Group forming prioritization

Use the Group prioritization operation to order records within each break group, which controls which records are used as the drivers during the comparison process.

Post-match prioritization

Add a Group prioritization operation before a Group Statistics operation to order records within a match group to control which record is flagged as the master record of each group of matching records. Add a Group prioritization operation before a Best Record operation to order records within a match group to control the destination of data that is being propagated from other records to form a best record.

Option	Description
Prioritization name	Specifies the name for this Group prioritization operation. If you have multiple operations in this Match transform, be sure to make this name unique.

Priority fields

Use the Priority fields table to order your break groups based on the content of a field (for example, dollar amount or date). Use the buttons to add, remove, and order rows. Place the primary sort field at the top of the list. The rest of the fields, in the order that they are positioned, determine the sub-sort that occurs.

Option	Description
Input field	Choose a field to sort your records on.
Field order	Specifies in which order records should be sorted.

3.5.4.5.3.1.5 Group prioritization options: Record Completeness tab

Option	Description
Prioritization name	Specifies the name for this Group prioritization operation. If you have multiple operations in this Match transform, be sure to make this name unique.
Order records based on completeness of data	Select this option to apply priority and blank penalty points to records to help control the order of your records.
Define only field penalties	Select this option when penalties need to be assessed based on blank fields.
Define priority and penalty fields	Select this option when you have specific fields that contain the actual integer values for priority and blank penalty.
Record priority field	Choose the field that contains priority values. This field must contain an integer.
Apply blank penalty field	Choose the field that contains the indicator (Y or N) for applying blank penalty points to a record.

Option	Description
Define priority and penalty based on input source	Select to have your record priority and blank penalty indicator (Y or N) determined by membership in a given source.
Source Name	Choose an input source from the drop-down list in the Source Name column. The sources listed here are defined in the Input Source operation.
Priority	Type a priority value (an integer) in the Priority column. Remember that the lower the priority score, the higher the priority.
Apply Blank Penalty	Choose Yes or No to determine whether a blank penalty is applied to a record based on membership to this source.
Default record priority	Specifies the default value for the record priority if the record does not contain a field with this value, this field is blank for a record, or if a record does not belong to any of the sources specified. Remember that the lower the priority score, the higher the priority.
Default apply blank penalty	Specifies the default indicator as to whether to add blank penalty points to records with blank fields. This indicator is used if a record does not have a field that carries this indicator, if that field is blank or has invalid data, or ifa record does not belong to any of the sources specified. *Yes:* Each record's blank penalty is added to the record's record priority to generate an adjusted record priority score. The lower the score, the higher the priority. *No:* No penalty is applied when the fields are blank.
Input field	Displays the input fields available to assign a blank penalty score to.
Blank penalty	Assign a penalty value (an integer) to apply when the specified field is blank in a record.

3.5.4.5.3.1.6 Unique ID options: Unique ID tab

Use the Unique ID options to assign sequential identification numbers to each new record when adding records to a data warehouse. For example, the largest number assigned in a particular project can be carried over as the beginning identification number (plus 1) to be used in the assignment of new sequential IDs. This occurs when the software processes the next source against the data warehouse file.

Note

Also see the Unique ID section for information about working with unique ID in a multi-server environment. Depending on the processing operation and starting value source you use, there could be limitations for using unique ID.

The Unique ID option group includes the following options:

Option	Description
Unique ID name	Enter a name for this Unique ID operation. If you are using other Unique ID operations in this Match transform, you may want to specify the name of the match transform and match level in this name to distinguish it from others.
Processing operation	Specifies the type of processing operation you want the application to perform. Valid values include: *Assign:* Assigns a new ID to unique records that need one, or assigns a new ID to all members of a group that don't have an ID. In addition, the assign operation copies an existing ID when a member of a match group already has an ID. For assign operations to work, all match group members must appear consecutively in one collection and must be in priority order (high to low). *AssignCombine:* Performs both an assign operation and a combine operation. All match group members must appear consecutively in one collection and must be in priority order (high to low). *Combine:* Combines the IDs of a match group when more than one ID is represented. All match group members must appear consecutively in one collection and must be in priority order (high to low). *Delete:* Removes unique IDs from records that have one, unless they are protected. *Split:* Splits the IDs of an ID group when more than one match group is represented. All ID group members must appear consecutively in one collection and must be in priority order (high to low).
Recycle unique IDs	Specifies whether unique IDs that were freed up during the delete operation should be used again in different records. You may want to recycle unique IDs if you have a limited amount available. Valid values include: *Yes:* Recycle freed-up unique IDs. *No:* Do not recycle freed-up unique IDs.
ID field	A field that holds a previously assigned unique ID. If this field is omitted, then it is assumed that no records have a unique ID.
Field	The starting unique ID is obtained from an input field. Be sure to map in a field from an upstream transform before you add this option.
Starting unique ID field	Choose the field that passes in the starting unique ID. If no Unique ID is received, the starting number will default to 1.
Constant value	The starting ID is specified as a positive whole number in the Starting value option.
Starting value	Indicates the starting unique ID value. Valid values range from 1 to UINT_MAX (unsigned integer max). The default value is 1.
Value from file	The starting Unique ID is read from the file specified in the File option.

Option	Description
File	Specifies the path and name of the file that manages unique IDs. A value is required here only when the Starting unique ID source option is set to File.
GUID	Uses the Globally Unique Identifier (GUID) as the unique ID. This is also known as the Universal Unique Identifier (UUID). The UUID variation used for unique ID is a time-based 36-character string with the format: `TimeLow-TimeMid-TimeHighAndVersion-ClockSeqAndReservedClockSeqLow-Node`. For more information about UUID, see the RFC document in the Related Topics section.
Save ending ID to file and reclaim recycled IDs	Specifies whether to save the last ID that was assigned to a file. Additionally, specifies whether to reclaim recycled IDs.
File	Specifies the file to write the last assigned ID to.
Allow multiple Match transforms to access unique ID file	Allows multiple Match transforms to access a shared unique ID file. When enabled, multiple data flows can access the same unique ID file, and single Match transforms can run in more than one process when the DOP setting is greater than 1. In addition, this allows multiple Match transforms within a single data flow to share a single unique ID file.
Number of IDs to get when accessing file	Specifies the number of IDs to retrieve from the unique ID file during each access. For example, with a setting of 100, the first process will access the file and retrieve IDs numbered 1-100. The next process with retrieve IDs numbered 101-200. If a process uses less than the number of retrieved IDs, the remaining IDs are written back to the unique ID file as recycled IDs. ℹ Note A setting greater than 1 improves performance when sharing a unique ID file between multiple processes by reducing the number of times the file must be accessed. However, integer numbers may not be assigned in sequential order.
Group number field	Specifies the field that holds a group number. The group number is used to assign the same unique ID to more than one record. If this field is omitted, then it is assumed that each record is unique and should have its own number.

Related Information

Designer Guide: Match, Assign unique IDs using a file [page 570]
Designer Guide: Match, Unique ID [page 570]

3.5.4.5.3.1.7 Unique ID options: Destination Protection tab

Use the Destination Protection tab to control whether a record's unique ID is protected based on the source that the record belongs to. This can help prevent IDs from being assigned to a suppression or rented source.

Option	Description
Unique ID name	Enter a name for this Unique ID operation. If you are using other Unique ID operations in this Match transform, you may want to specify the name of the match transform and match level in this name to distinguish it from others.
Enable destination protection	Select if you want to protect a destination source from having its unique IDs overwritten with the IDs from matching records.
Default destination protection	Select the default destination protection setting. This is useful because the default setting will account for records that are protected (or not protected) through the use of sources or fields.
Specify destination protection by field	Select to enable destination protection through a value in a field.
Unique ID protected field	Choose an input field from the drop-down list that holds the value for specifying whether this ID is protected. The field must contain a value of Y or N. Any other value (including blank) will cause the default destination protection specification to occur, if you specified one.
Specify destination protection by source	Select this option to control destination protection through membership in a particular source. Fill in the table with source names and whether they are protected.
Source name	Choose the name of the source from the drop-down list. The list here is populated with defined sources and source groups from the *Input Sources Editor* window of the Match Editor.
Unique ID protected	*Yes:* This source is protected. *No:* This source is not protected.

3.5.4.5.4 Output fields

The Associate transform requires that you map one field on output: Group_Number.

Field	Default content type	Description
Group_Number	Group_Number	The group number resulting from the association process. Records that belong to the same match group share the same group number. The group numbers start with the number one. Unique records have a blank group number.

Group prioritization output fields

The following output fields are available when you add a Group Prioritization operation to an Associate transform

Field	Description
Priority_Value	Record priority value assigned to record. If you did not include a priority value, this field outputs 0.

3.5.4.6 Country ID

The Country ID transform parses your input data and then identifies the country of destination for each record. After identifying the country, the transform can output the country name, any of three different ISO country codes, an ISO script code, and a percentage of confidence in the assignment.

Though you can use the Country ID transform before any transform in a data flow, you will probably find it most useful during a transactional address cleanse job. Place the Country ID transform before the Global Suggestion List transform. The Global Suggestion List transform needs the ISO_Country_Code_2Char field that the Country ID transform can output.

It is not necessary to use the Country ID transform before the Global Address Cleanse transform in a data flow because the Global Address Cleanse transform contains its own Country ID processing. It is also not necessary to use the Country ID transform before the USA Regulatory Address Cleanse transform because your input data should contain U.S. addresses only.

Country ID transform configuration

Data Quality provides you with a sample transform that can help you get started creating a Country ID transform useful to you.

Related Information

Input Fields [page 1147]
Output fields [page 1147]
Transform configurations [page 1123]

3.5.4.6.1 Input Fields

Use the Input_Fields option group to map the input field that you want to use in this transform.

Here is a list of the Country ID input fields and their descriptions.

Field	Description
Country	The country's name or code.
Lastline	The locality, region, and postal code on one line.
Locality1-3	City, town, or suburb information.
Multiline1-12	Lines that may contain any data. The type of data in these lines may vary from record to record.
Postcode	The postal code for the address.
Region1	The state, province, territory, or region of the address.

3.5.4.6.2 Output fields

The following are Data Service output fields that can be defined in the Output tab of the transform.

Field	Description
Confidence_Score	The percentage of certainty that the identified country is accurate. For example, a value of 100 is 100% certainty.
Country_ID_Info_Code	The Country ID info code when the Country ID transform cannot determine a country. *1010*: Indicates a tie in identifying the country. *1005*: Indicates that no country was identified.
Country_Name	The identified country name.
ISO_Country_Code_2Char	The 2-character ISO code for the identified country.
ISO_Country_Code_3Char	The 3-character ISO code for the identified country.
ISO_Country_Code_3Digit	The 3-digit ISO code for the identified country.
ISO_Script_Code	The 4-character script code to use for the identified country, such as LATN or KANA.

Related Information

Input Fields [page 1147]

3.5.4.7 Data Cleanse

Use the Data Cleanse transform to parse and format custom or person and firm data as well as phone numbers, dates, e-mail addresses, and Social Security numbers. Custom data includes operational or product data specific to your business.

> Note
>
> The Social Security number algorithm is no longer updated by the Social Security Administration. The SSN validation performed in Data Cleanse is based on data through July, 2011. The Social Security data is built into the cleansing package.

The cleansing package you specify defines how your data should be parsed and standardized.

Within a data flow, the Data Cleanse transform is typically placed after the address cleansing process and before the matching process.

3.5.4.7.1 Content objects

Transform configurations

A transform configuration is a transform with preconfigured input fields, output fields, and options that can be used in multiple data flows. These are useful if you repeatedly use a transform with specific options and input and output fields.

When Data Services is installed, read-only transform configurations are provided for the Data Quality transforms.

You can use transform configurations in your data flows or as an example of a typical transform. After you place an instance of the transform configuration in a data flow, you can override these preset defaults. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one.

Sample blueprints and other objects

We have created Data Quality blueprints and other content objects to help you set up Data Services jobs. We've identified a number of common scenarios that you are likely to perform with Data Services. For each scenario, we've included a blueprint that is already set up to solve the business problem in that scenario.

Related Information

Downloading blueprints and other content objects [page 1128]
Transform configurations [page 1123]

3.5.4.7.2 Data Cleanse options

The Data Cleanse transform includes options that control how person, firm, and custom data are parsed and standardized.

3.5.4.7.2.1 Common options

Option	Description
Run As Separate Process	*Yes:* Splits the transform into a separate process. *No:* Keeps the transform in the same process as the rest of the data flow.

3.5.4.7.2.2 Report and analysis

Use this option to generate report data for the Data Cleanse transform.

Option	Description
Generate Report Data	Specifies whether to generate report data for this transform. *Yes*: Generates report data for this transform. *No*: Turns off report data generation. If you do not need to generate reports (during testing, for example), you should set this option to *No* to improve performance.

3.5.4.7.2.3 Cleansing Package options

Controls the cleansing package options the Data Cleanse transform uses.

Option	Description
Cleansing Package Name	Selects the cleansing package that the Data Cleanse transform will use to parse data.

Option	Description
Content Domain Sequence	A content domain specifies which domain's properties should be assigned to a variation. You can specify more than one content domain. The Global domain is a special content domain which contains all variations and their associated properties. If a variation is not associated with domain-specific information the Global domain serves as the default domain. The Global domain is required for every content domain sequence. Be sure to add GLOBAL as the last domain in the sequence. ℹ Note You can set this option as a dynamic input field. Select the content domains you want to include. The arrows allow you to change the order of the content domains. GLOBAL - Global AR - Arabic ZH - Chinese CS - Czech DA - Danish NL - Dutch EN_US - English (United States & Canada) EN_GB - English (United Kingdom & Ireland) EN_AU - English (Australia & New Zealand) EN_IN - English (India) FR - French DE - German HU - Hungarian ID - Indonesian IT - Italian JA - Japanese MS - Malay NO - Norwegian PL - Polish PT_BR - Portuguese (Brazil) PT_PT - Portuguese (Portugal) RO - Romanian

Option	Description
	RU - Russian SK - Slovak ES_MX -Spanish (Latin America) ES_ES - Spanish (Spain) SV - Swedish
Output Format	Selects the format for output. Based on the specified domain in the output format, Data Cleanse uses certain output fields and formats the data in those fields according to the regional standards. Note You can set this option as a dynamic input field. Valid values for this option are: *AR* Arabic *ZH* Chinese *CS* Czech *DA* Danish *NL* Dutch *EN_US* English (United States & Canada) *EN_GB* English (United Kingdom & Ireland) *EN_AU* English (Australia & New Zealand) *EN_IN* English (India) *FR* French *DE* German *HU* Hungarian *ID* Indonesian *IT* Italian *JA* Japanese *MS* Malay *NO* Norwegian *PL* Polish *PT_BR* Portuguese (Brazil) *PT_PT* Portuguese (Portugal)

Option	Description
	RO Romanian *RU* Russian *SK* Slovak *ES_MX* Spanish (Latin America) *ES_ES* Spanish (Spain) *SV* Swedish

For more information, see the topics “About domains” and “About output format” in the *Information Steward User's Guide*.

3.5.4.7.2.4 Options

The Options group includes settings that control how the Data Cleanse transform parses and outputs data.

Option	Description
Filter Output Fields	Specifies which output fields are displayed in the Output tab. *Show_Relevant_Fields:* The fields available in the Output tab are based on the mapped input fields and the selected parser sequence multiline options. Includes all output fields that could possibly contain parsed data. The **`<Extra>`** fields are always available. *Show_All_Fields:* All Data Cleanse transform fields are available.

3.5.4.7.2.5 Input word breaker

Controls how the parser breaks input data.

Option	Description
Break on Whitespace Only	Specifies whether the Data Cleanse transform breaks input data only on white space or on white space, punctuation, alphanumeric transitions, and script transitions. *Yes:* Input data breaks only on white space. *No:* Input data breaks on white space, punctuation, alphanumeric transitions, and script transitions. For data in the CJK and Kana scripts, input data breaks on each character.

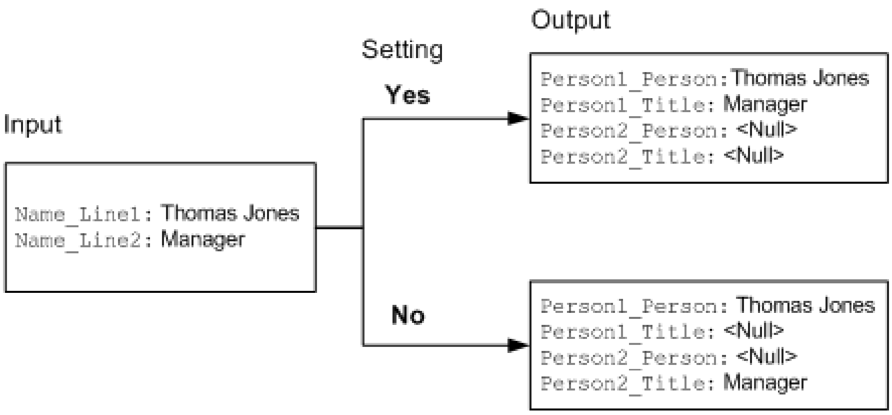
Option	Description
	This option allows the Data Cleanse transform to recognize alphanumeric product codes as entries in a custom cleansing package. For example, if *Break on Whitespace only* is set to no, the parser breaks a product code such as AF302 into two tokens, AF and 302. If *Break on Whitespace only* is set to yes, the parser recognizes AF302 as a single entry. Note This option typically applies to custom cleansing packages. The out of the box person and firm cleansing packages are designed to use the parsing strategy that breaks data on white space, punctuation, alphanumeric transitions, and script transitions.

3.5.4.7.2.6 Person standardization options

Controls how the Data Cleanse transform standardizes person-related output.

> Note
>
> The options in this group apply only to person and firm cleansing packages.

Option	Description
Assign Prenames	Specifies whether the transform should include assigned prenames (for example, Mr. or Mrs.) in the Prename output field. The Prename output field always includes prenames that are part of the name input data. Additionally, the Data Cleanse transform can assign prenames based on the gender of the name (strong_male or strong_female) in the Given_Name1 field. When the gender of Given_Name1 is not strong, prenames are assigned based on the settings for the *Gender Options* > *Use Given Name2 To Assign Gender* and *Gender Options* > *Use Family Name to Assign Gender* options. *Yes:* Turns on prename assignment. *No:* Turns off prename assignment. The Prename output field contains only prenames included in the input data
Associate Name Title	Defines how name and occupational title data found in separate input fields are associated. *Yes:* Data Cleanse assumes that the name and title data describe the same person and is associated. *No:* Data Cleanse assumes that the name and title data is not associated.

Option	Description
	For example, the diagram below shows the difference in the output based on the *Associate Name Title* setting.  Name_Line1: Thomas Jones Name_Line2: Manager Person1_Person:Thomas Jones Person1_Title: Manager Person2_Person: <Null> Person2_Title: <Null> Person1_Person: Thomas Jones Person1_Title: <Null> Person2_Person: <Null> Person2_Title: Manager
Combine Compound Names	Specifies how compound family names are standardized when the family name includes a Pre_Family_Name that is also a Pre_Family_Name_Combine. *Yes:* Combines compound family names. For example, the family name Mc Donald would combine to McDonald. *No:* Retains a space between the Pre_Family_Name and the family name. For example, the family name Mc Donald remains Mc Donald. Note This option has no impact on a Pre_Family_name that is not classified as a Pre_Family_Name_Combine. For example, the name Van Helsing will not be combined because "Van" is not classified with both PRE_FAMILY_NAME and PRE_FAMILY_NAME_COMBINE.
Enable Presumptive Name Parsing	Specifies whether you want to use presumptive name parsing on Name_Line input fields. *Yes:* Turns on presumptive name parsing. Data in the Name_Line input field is treated as a name. *No:* Turns off presumptive name parsing. Data in the Name_Line input field that does not parse as a name remains unparsed and is output to the Extra field. For example, if the data contains an automobile brand and model in a Name_Line input field, the Data Cleanse transform tries to parse the information as a name based on rules in the cleansing package. If the option is set to No and Data Cleanse is not able to assign the data, the unparsed data is output to the Extra field. If the option is set to Yes, Data Cleanse will assign the data as a name.
Name Order	Defines how Data Cleanse applies parsing rules to determine the content of the Given_Name and Family_Name output fields.

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><p><em>Given_Family_Name_Strict</em> and <em>Family_Given_Name_Strict</em>: These values specify the respective order of given and family names in the input file. Parsing rules that do not follow the strictly-defined name order are not considered when Data Cleanse determines which rule to apply to the input string.</p>
<p>These settings are useful when the order of the family and given names in the input data is consistent.</p>
<p><em>Given_Family_Name_Suggest</em> and <em>Family_Given_Name_Suggest</em>: These values specify which rule to choose in order to break a tie when two rules have the same confidence score. Data Cleanse chooses the rule that follows the suggested name order.</p>
<p><em>Unknown</em>: Data Cleanse chooses the rule with the highest confidence score based on information in the cleansing package. In the case of a tie, Data Cleanse chooses the first rule in the rule order.</p></td></tr>
<tr><td>Parse Discrete Input</td><td><p>Defines how to parse person data.</p>
<p><em>No</em>: discrete input fields are mapped directly to the corresponding output fields without being parsed.</p>
<p><em>Yes</em>: Discrete input fields are combined to one input field so the data can be parsed and output to discrete fields.</p>
<p>Example:</p>
<p>Table 156: Input data</p>
<table>
<tr><th>Column</th><th>Field</th></tr>
<tr><td>Person1_Given_Name1</td><td>Mr John T</td></tr>
<tr><td>Person1_Family_Name1</td><td>Smith Iii</td></tr>
</table>
<p>Table 157: Output data</p>
<table>
<tr><th>Column</th><th>Option=No</th><th>Option=Yes</th></tr>
<tr><td>Person1.Prename</td><td>&lt;blank&gt;</td><td>Mr</td></tr>
<tr><td>Person1.Given_Name1</td><td>Mr John T</td><td>John</td></tr>
<tr><td>Person1.Given_Name2</td><td>&lt;blank&gt;</td><td>T</td></tr>
<tr><td>Person1.Family_Name1</td><td>Smith Iii</td><td>Smith</td></tr>
<tr><td>Person1.Maturity_Postname</td><td>&lt;blank</td><td>III</td></tr>
</table></td></tr>
</table>

3.5.4.7.2.7 Gender standardization options

The gender standardization options control which input fields Data Cleanse uses to assign gender. These options are found in the *Gender Options* group under ▶ *Options* ▸ *Standardization Options* ▸ *Person* ◢.

ℹ Note

The options in this group apply only to person and firm cleansing packages.

Option	Description
Use Given Name2 To Assign Gender (Gender Options)	When the gender of the prename and Given_Name1 are unassigned or ambiguous, assigns gender based on the gender of the parsed Given_Name2. *Yes:* Turns on the option. *No:* Turns off the option. For example, if the option is set to No, the gender of the name Pat Robert Smith is ambiguous because the Given_Name1, Pat, is ambiguous. However, if the option is set to Yes, the gender is Strong_Male because the Given_Name2, Robert, is Strong_Male. The same logic applies if the name were P. Robert Smith; the Given_Name1, P, is ambiguous.
Use Family Name To Assign Gender (Gender Options)	When the gender of the prename, Given_Name1, and Given_Name2 are unassigned or ambiguous, assigns gender based on the gender of the family name. Uses Family_Name1 if gender is assigned and is not ambiguous. Uses Family_Name2 if unable to use Family_Name1. *Yes:* Turns on the option. *No:* Turns off the option. For example, if the option is set to No, the gender of the name N. Albiantsev is ambiguous because the Given_Name1, N., is ambiguous. However, if the option is set to Yes, the gender is Strong_Male because the Family_Name1, Albiantsev is Strong_Male.

3.5.4.7.2.8 Firm standardization options

The firm standardization options control how the Data Cleanse transform standardizes firm-related output.

ℹ Note

The options in this group apply only to person and firm cleansing packages.

Option	Description
Enable Presumptive Firm Parsing	Specifies whether you want to use presumptive firm parsing on Firm_Line input fields. *Yes*: Turns on presumptive firm parsing. Data in the Firm_Line input field is treated as a firm name. *No*: Turns off presumptive firm parsing. Data in the Firm_Line input field that does not parse as a firm remains unparsed and is output to the Extra field.

Option	Description
	For example, if the data has a given name and family name in a Firm_Line input field, the Data Cleanse transform tries to parse the information as a firm based on rules in the cleansing package. If the option is set to No and Data Cleanse is not able to assign the data, the unparsed data is output to the Extra field. If the option is set to Yes, Data Cleanse will assign the data as a firm.

3.5.4.7.2.9 Other standardization options

Standardization options control how the Data Cleanse transform standardizes many types of output.

Option	Description
Capitalization	Specifies the casing of your output. *Lower*: Converts the output to lowercase. For example, john mckay. *Mixed*: Preserves the casing for the standard form as defined within the cleansing package. If a standard form is not defined, the output is converted to mixed case. For example, if the standard form is defined as John Mckay, that would be preserved. If a standard form is not defined, the output is converted to mixed case, John McKay. *Preserve*: Preserves the input casing. *Upper*: Converts the output to uppercase. For example, JOHN MCKAY.
Character Width Style	Specifies the character width used in output fields. Useful when processing Japanese or mixed language data. *Normal_Width*: Output field width reflects the normalized character width based on the script type. Thus some output columns contain half-width characters and other columns contain full-width characters. For example, all full-width Latin characters are standardized to their half-width forms and all half-width katakana characters are standardized to their full-width forms. Normal_Width does not require special processing and therefore is the most efficient setting. *Full-width*: Characters are converted from their half-width forms to full-width forms for all output fields. For characters that do not have full-width forms, the half-width forms are used. *Half-width*: Characters are converted from their full-width forms to half-width forms for all output fields. For characters that do not have half-width forms, the full-width forms are used.

Option	Description
	ℹ Note Since the output width is based on the normalized width for the character type, the output data may be larger than the input data. You may need to increase the column width in the target table. For template tables, selecting the *Use NVARCHAR for VARCHAR columns in supported databases* box changes the VARCHAR column type to NVARCHAR and allows for increased data size.
North American Phone Delimiter	Specifies a character to use as a delimiter for phone output following the North American Numbering Plan (NANP). *Backslash (\)*: Uses backward slashes as the delimiter in the phone number. For example, 123\656\5000. *Dash (-)*: Uses dashes as the delimiter in the phone number. For example, 123-656-5000. *Slash (/)*: Uses forward slashes as the delimiter in the phone number. For example, 123/656/5000. *None*: Does not add a delimiter to the phone number. For example, 1236565000. *Period (.)*: Uses periods as the delimiter in the phone number. For example, 123.656.5000. *Space*: Uses spaces as the delimiter in the phone number. For example, 123 656 5000.
North American Phone Delimiter After Area	Specifies placement of a delimiter between the area code and prefix phone output following the North American Numbering Plan (NANP). To use this option, you must also specify a delimiter in the North American Phone Delimiter option. *Yes*: Adds a delimiter. For example, 123-656-5000. *No*: Does not add a delimiter. For example, 123 656-5000.
North American Phone Parens Area	Controls placement of parentheses () around the area code of phone number output following the North American Numbering Plan (NANP). *Yes*: Includes the parentheses. For example, (123) 656-5000. *No*: Omits the parentheses. For example, 123 656-5000.
One-to-one mapping	Specifies whether to place the input data into the corresponding output field for the following parsers: Phone, Email, Date. *Yes*: Places the parsed data into the corresponding output field. For example, if on input Date1 and Date2 are blank and Date3 contains data, then on output, Date1 and Date2 are blank and the data is placed in Date3.

Option	Description
	No: Places the parsed data into the first available output field in the category. For example, if on input Date1 and Date2 are blank and Date3 contains data, then on output, Date1 contains the parsed data that was input in the Date3 field.
Phone Extension Text	Specifies the standard text for a phone extension. For example, Ext.
Remove Diacritical Characters	Removes diacritical characters and replaces it with the ASCII equivalent. *Yes*: Replaces diacritical characters such as accent marks, umlauts, and so on with the ASCII equivalent. *No*: Retains the standardized diacritical characters. For example, when the option is set to *No*, the data is output with accent marks such as `María Hernández` or `Geschäftsführer`. When the option is set to *Yes*, the data is output without accent marks such as `Maria Hernandez` or `Geschaeftsfuehrer`.
Remove Punctuation	Removes all punctuation from standardized data (with the exception of hyphens between names). *Yes*: Removes punctuation. *No*: Leaves the punctuation as is on input. For example, if the standard form for extra large is X.L. and the option is set to Yes, the standardized output becomes XL.
SSN Delimiter	Specifies which character to use for standard U.S. Social Security number (SSN) output delimiters. *Backslash (\)*: Uses backward slashes as the delimiter in the SSN. For example, 799\45\6789. *Dash (-)*: Uses dashes as the delimiter in the SSN. For example, 799-45-6789. *Slash (/)*: Uses forward slashes as the delimiter in the SSN. For example, 799/45/6789. *None*: Does not add a delimiter to the SSN. For example, 799456789. *Period (.)*: Uses periods as the delimiter in the SSN. For example, 799.45.6789. *Space*: Uses spaces as the delimiter in the SSN. For example, 799 45 6789.

3.5.4.7.2.10 Date options

Configures standards for date-related data.

Option group	Description
Century Threshold	Indicates whether a two-digit date is considered part of the 20th or 21st century. The default value is 25. Specify a two-digit integer that represents the first year that a parsed two-digit year is considered part of the 21st century (20xx). All two-digit years greater than the specified integer are considered part of the 20th century (19xx). For example, if you enter 11, all two-digit years 11 or lower are considered part of the 21st century. 08 is considered 2008. 11 is considered 2011. All two-digit years higher than 11 are considered part of the 20th century. 12 is considered 1912.
Date Delimiter	Specifies what character to use for standard date output delimiters. *Backslash (\)*: Uses backward slashes as the delimiter for the date. For example, 04\01\2010. *Dash (-)*: Uses dashes as the delimiter for the date. For example, 04-01-2010. *Slash (/)*: Uses forward slashes as the delimiter for the date. For example, 04/01/2010. *None*: Does not add a delimiter to the date. For example, 04012010 *Period (.)*: Uses periods as the delimiter for the date. For example, 04.01.2010. *Space*: Uses spaces as the delimiter for the date. For example, 04 01 2010. *Chinese_Japanese*: Uses the following Chinese/Japanese characters as delimiters: ● 月 always follows the month ● 日 always follows the day ● 年 always follows the year An example of Arabic numbers with Chinese/Japanese delimiters is: 04 月 01 日 2010 年 An example of Chinese/Japanese Numbers with Chinese/Japanese delimiters is: 四 月 一 日 二千零一 十 年
Date Format	Specifies how to standardize date output. YEAR_MONTH_DAY: For Example, 2012-08-16 YEAR_DAY_MONTH: For Example, 2012-16-08 MONTH_DAY_YEAR: For Example, 08-16-2012 DAY_MONTH_YEAR: For Example, 16-08-2012
Enable Zero Pad	Specifies placement of a zero on the front of one-digit days and months. For example, July 4 could be 04-07 (or 07-04) with a zero pad, and 4-7 (or 7-4) without a zero pad. *Yes*: Turns on the option. *No*: Turns off the option.

Option group	Description
Input Month Before Day	Specifies whether the date follows the pattern of having the month first or the day first in the input. *Yes:* The month is first. For example, 11/12/2004 would be November 12, 2004. *No:* The day is first. For example, 11/12/2004 would be December 11, 2004.
Input Year First	Specifies whether the date follows the pattern of having the year first in the input. *Yes*: The year is first. For example, if your input is 03/02/04, the transform will convert it to 2003 February 4. *No*: The month is first. For example, 03/02/04 would be March 2, 2004.
Month Format	Specifies how to standardize date and month components. *Full_Text*: Standardizes output with spelled-out months. The language of the month is based on the domain selected in the Output Format option. For example, if one of the English domains is selected, the full-text month would be January, February, March, and so on. *Numeric*: Standardizes output with numeric months (for example, 03). *Short_Text*: Standardizes output with abbreviated months. The language of the month is based on the domain selected in the Output Format option. For example, if one of the English domains is selected, the short-text month would be Jan., Feb., Mar., and so on.
Numeric Format	Specifies the format of numeric date values *Arabic_Numbers*: Returns numeric date values in Arabic *Chinese_Japanese_Numbers*: Returns numeric date values in Chinese or Japanese.
Year Format	Specifies how to standardize date and year components. *Full:* Standardizes output with four-digit years (for example, 2004). *Short:* Standardizes output with two-digit years (for example, 04).

3.5.4.7.2.11 Parser configuration

Controls which parsing engines Data Cleanse uses for parsing multiline fields and the order in which they are applied. If a particular parser is not included, Data Cleanse does not look for that type of data in the input field.

Option	Description
Parser Sequence Multiline1-12	*Custom Parser:* Parses custom operational or product data for the category in the specified custom cleansing package. *Date:* Parses data as a date. *Email:* Parses data as an e-mail address.

Option	Description
	Firm: Parses data as firm name. *Person:* Parses data as a personal name *Person or Firm:* Parses data as a personal or firm name. *Phone:* Parses data as phone numbers. Any non-North American numbers are parsed first and can be output to the Phone and International_Phone fields. North American numbers are parsed later using thing North American Numbering Plan (NANP). The data can be output to Phone and North_American_Phone output fields. Phone number patterns that do not follow the NANP must be defined in Cleansing Package Builder. *SSN:* Parses data as a U.S. Social Security number. *UDPM:* Parses data using user-defined patterns created in Cleansing Package Builder.

3.5.4.7.2.12 Memory cache

To improve performance, you can set the cache in kilobytes (KB) to allocate a certain amount of memory for use in the Data Cleanse transform. For example, if you have a 39 MB cleansing package, you can set the cache size to 40000 KB and use the rest of available memory for processing other transforms. Likewise, if your cleansing package is larger, you can increase the value.

Option	Description
Memory in KB for Cache	(Optional) Enter a value of memory to allocate per thread for processing the Data Cleanse transform. Setting this option to `0` indicates that the internal default value (65536) should be used.

3.5.4.7.3 Input fields

The following are recognized input fields that you can use in the input mapping for the Data Cleanse transform. The fields are listed alphabetically.

Name	Description
Date1-6	Date. For example, 08/16/2004.
Email1-6	E-mail address.
Firm_Line1-2	Firm name, firm location, or both.
Firm_Location1-2	Location within a company or organization, such as a department, mail stop, room, or building.
Firm_Name1-2	Name of a company or organization.

Name	Description
Multiline1-12	Multiline data. Item types from this input are parsed in the order set in the Parser Sequence Multiline option, including parsers from custom cleansing packages.
Name_Line1-6	Whole name or names. May include job title.
Name_Or_Firm_Line1-6	Name of a person or organization.
Option_Country Option_Language Option_Region	The content domain sequence and output format usually can be automatically generated based on the Option_Country data. However, there are a few countries where Option_Language and Option_Region data is helpful to make the assignment, for example, Switzerland, Belgium, and Canada. Using Option_Language and Option_Region is optional, and is only used to determine the most appropriate content domain and output format. These input fields should be mapped from the following Global Address Cleanse output fields in this order: ● ISO_Country_Code_2Char ● Language ● Region1 These three input fields work together to determine the Option_Content_Domain_Sequence and/or Option_Output_Format.
Option_Content_Domain_Sequence (Dynamic input field)	The content domain sequence. The valid predefined values are: AR, CS, DA, DE, EN_AU, EN_GB, EN_IN, EN_US, ES_MX, ES_ES, FR, HU, ID, IT, JA, MS, NL, NO, PL, PT_BR, PT_PT, RO, RU, SK, SV, ZH, and GLOBAL.
Option_Output_Format (Dynamic input field)	The format for output specified as an abbreviation of the domain. The valid predefined values are: AR, CS, DA, DE, EN_AU, EN_GB, EN_IN, EN_US, ES_MX, ES_ES, FR, HU, ID, IT, JA, MS, NL, NO, PL, PT_BR, PT_PT, RO, RU, SK, SV, and ZH.
Person1_Family_Name1 Person2_Family_Name1	Discrete family name (for example, Smith).
Person1_Family_Name2 Person2_Family_Name2	Second discrete family name. May be useful for cultures where people are known by both paternal and maternal family names. If your input data contains two family name fields, map the first to Person1_Family_Name1 and the second to Person1_Family_Name2.
Person1_Given_Name1-2	Discrete given names (for example, John or B.).

Name	Description
Person2_Given_Name1-2	
Person1_Honorary_Postname Person2_Honorary_Postname	Honorary postname indicating certification, academic degree, or affiliation, such as CPA.
Person1_Maturity_Postname Person2_Maturity_Postname	Maturity postname indicating heritage, such as Jr., Sr., III.
Person1_Prename Person2_Prename	Discrete prename, such as Mr., Mrs., Dr., or Lt. Col.
Person1_Title Person2_Title	Discrete job title, such as Software Engineer.
Phone1-6	Phone number. Data Cleanse will first try to parse the number as an international phone number. If that fails, it will try to parse it as a North American phone number.
SSN1-6	U.S. Social Security number.
Title_Line1-6	Job title (for example, Accountant).
UDPM1-4	Input field associated with patterns and rules defined in the User-defined type of Reference Data in Cleansing Package Builder. For example, CN244-56.

Related Information

Dynamic transform settings [page 1129]

3.5.4.7.4 Output fields

The following are recognized output fields that you can use in the output mapping for the Data Cleanse transform. By default, the Extra, Info_Code, and Status_Code fields are always available, additional output fields are displayed based on the mapped input fields and the selected parser sequence multiline options. Additionally, the Ignore field may be available.

You can use the *Filter Output Fields* option to display a complete list of output fields. The fields are listed alphabetically.

Generated field name	Content type	Description
Date	Date	A date that is parsed.
Date_Day	Date	The day that is parsed from the date.

Generated field name	Content type	Description
Date_Month	Date	The month that is parsed from the date.
Date_Year	Date	The year that is parsed from the date.
Dual_Name	None	Set of components resulting from one input field that contains two names separated by a connecting word such as "and" or "or." Example 1: Input: Terry and Kris Johnson Output: Terry Johnson and Kris Johnson Example 2: Input: Terry Johnson or Kris Adams Output: Terry Johnson or Kris Adams
Email	Email	An entire e-mail address.
Email_Domain_All	Email	The domain of the e-mail address. For example, sap.com.
Email_Domain_Fifth	Email	In an e-mail address with more than one domain listed, this field parses the fifth to last domain.
Email_Domain_Fourth	Email	In an e-mail address with more than one domain listed, this field parses the fourth to last domain.
Email_Domain_Host	Email	The host of the e-mail address (the first item listed after the @ symbol). For example, in "joex@sap.com", "sap" is returned.
Email_Domain_Second	Email	In an e-mail address with more than one domain listed, this field parses the second to last domain.
Email_Domain_Third	Email	In an e-mail address with more than one domain listed, this field parses the third to last domain.
Email_Domain_Top	Email	The last listed domain of the e-mail address. For example, .com.
Email_Is_ISP	Email	The email address is a known ISP (internet service provider) or email domain name listed in the email data type of Reference Data in Cleansing Package Builder.
Email_User	Email	The user name of the e-mail address. For example, in "joex@sap.com", "joex" is returned.

Generated field name	Content type	Description
Extra	None	Any data that is not parsed by any of the active parsers and thus Data Cleanse does not recognize the data as fitting one of the other output fields.
Family_Name1	Family_Name	Family name (for example, Smith).
Family_Name1_Match _Std1-6	Family_Name1_ Match_Std	The match standard for family names. This field is only used with cleansing packages that include name data in more than one script. The match standards include the name as it is written in alternate script types. For example, for a family name included in the Japanese dictionary in Kanji script, the match standards include Kana renditions of the name. If the dictionary does not have an alias entry, the output field is empty. Compare with Match_Family_Name.
Family_Name2	Family_Name 2	Second family name. May be used to output the paternal and maternal family names to separate fields.
Family_Name2_ Match_Std1-6	Family_Name2_ Match_Std	The match standard for second family names. This field is only used with cleansing packages that include name data in more than one script. The match standards include the name as it is written in alternate script types. For example, for a family name included in the Japanese dictionary in Kanji script, the match standards include Kana renditions of the name. If the dictionary does not have an alias entry, the output field is empty. Compare with Match_Family_Name.
Firm	Firm	The name of a company or organization.
Firm_Location	Firm_Location	A location within a company or organization, such as a department. For example, Mailstop.
Firm_Match_Std1-6	Firm_Match_Std	The match standard for firms. For example, HP is the match standard or alias for Hewlett Packard. If the cleansing package does not have an alias entry, the output field is empty.

Generated field name	Content type	Description
		Compare with Match_Firm.
Firm_Location_ Match_Std1-6	Firm_Location_ Match_Std	The match standard for firm locations. For example, MS is the match standard or alias for mailstop. If the dictionary does not have an alias entry, the output field is empty.
Gender	None	The gender description. The following output is available: *Ambiguous:* The name does not reliably indicate a gender. The name could be either male or female. For example, Pat. *Male_Strong:* High confidence that the person is male. That is, the name belongs to someone who is almost certainly a male. For example, John. *Male_Weak:* Some confidence that the person is male. That is, the name belongs to someone who is probably male. For example, Terry. *Female_Strong:* High confidence that the person is female. That is, the name belongs to someone who is almost certainly a female. For example, Mary. *Female_Weak:* Some confidence that the person is female. That is, the name belongs to someone who is probably a female. For example, Lynn. For dual names, the following output is also available: *Multi_Names_Ambiguous:* At least one of the names does not reliably indicate a gender. For example, Pat and John. *Multi_Names_Female:* Some or high confidence that both of the names belong to people who are female. For example, Mary and Lynn. *Multi_Names_Male:* Some or high confidence that both of the names belong to people who are male. For example, John and Terry. *Multi_Names_Mixed:* Some or high confidence that one of the names belongs to a person who is female, and the other name belongs to a

Generated field name	Content type	Description
		person who is male. For example, Lynn and John.
Gender_ID	None	A numeric value that corresponds to the gender description: *0:* Unassigned *1:* Male_Strong *2:* Male_Weak *3:* Ambiguous *4:* Female_Weak *5:* Female_Strong *6:* Multi_Names_Mixed *7:* Multi_Names_Male *8:* Multi_Names_Female *9:* Multi_Names_Ambiguous
Given_Name1	Given_Name	Given name (for example, Robert).
Given_Name1_ Match_Std1-6	Given_Name1_ Match_Std	The match standard for given names. For example, the application can tell you that Patrick and Patricia are potential matches for the given name Pat. Match standards can help you overcome two types of matching problems: alternate spellings (Catherine and Katherine) and nicknames (Pat and Patty). Compare with Match_Given_Name1.
Given_Name2	Given_Name2	Second given name.
Given_Name2_ Match_Std1-6	Given_Name2_ Match_Std	The match standard for second given names. For example, the application can tell you that Patrick and Patricia are potential matches for the given name Pat. Compare with Match_Given_Name2.
Honorary_Postname	Postname	Honorary postname indicating certification, academic degree, or affiliation. For example, CPA.
Honorary_Postname_Match_Std1-6	Postname_Match_Std	The match standard for an honorary postname. For example, M.B.A. is the match standard or alias for MBA.

Generated field name	Content type	Description
		If the dictionary does not have an alias entry, the output field is empty.
Info_Code	None	The code that identifies the rows that may require manual review because the data is suspect.
International_Phone	Phone	The entire international phone number, including extra items such as the country code.
International_Phone_ Country_Code	Phone	The country code of an international phone number.
International_Phone_ Country_Name	Phone	The name of the country of origin of an international phone number.
International_Phone_ Line	Phone	The portion of the international phone number that is not the country code or the city code.
International_Phone_ Locality_Code	Phone	The locality code of an international phone number.
Match_Family_Name	Family_Name1	The combined standardized form of Family-Name1 and FamilyName2 with a space between used in the Match transform during the comparison process. Data is output in uppercase, apostrophes are removed, and other punctuation is replaced with a single space. PreFamilyName data is removed. Compare with Family_Name1_Match_Std1-6 and Family_Name2_Match_Std1-6.
Match_Firm	Firm	A form of Firm that may be used in the Match transform during the comparison process. Data is output in uppercase, apostrophes are removed, other punctuation is replaced with a single space, and data that is extraneous for matching purposes is removed. This extraneous data includes business types such as Ltd. and GambH, and noise words such as The, And, and Of. Note Some words are classified to be removed from all domains, while others are language-specific and are classified to be removed in specific cultural domains. Compare with Firm_Match_Std1-6.

Generated field name	Content type	Description
Match_Given_Name1	Given_Name1	The standardized form of GivenName1 used in the Match transform during the comparison process. Data is output in uppercase, apostrophes are removed, and other punctuation is replaced with a single space. PreGivenName data is removed. Compare with Given_Name1_Match_Std1-6.
Match_Given_Name2	Given_Name2	The standardized form of GivenName2 used in the Match transform during the comparison process. Data is output in uppercase, apostrophes are removed, and other punctuation is replaced with a single space. PreGivenName data is removed. Compare with Given_Name2_Match_Std1-6.
Match_Maturity_Postname	Postname	The standardized form of MaturityPostname used in the Match transform during the comparison process. Data is output in uppercase, apostrophes are removed, and other punctuation is replaced with a single space. Compare with Maturity_Postname_Match_Std1-6.
Match_Phone	Phone	The form of Phone used in the Match transform during the comparison process. Data is output as a string of digits. Spaces, punctuation, alphabetical characters, and leading zeros are removed.
Match_Prename	Prename	The standardized form of Prename used in the Match transform during the comparison process. Data is output in uppercase, apostrophes are removed, and other punctuation is replaced with a single space. Compare with Prename_Match_Std1-6.
Maturity_Postname	Postname	Maturity postname indicating heritage, such as Jr., Sr., III.
Maturity_Postname_Match_Std1-	Postname_Match_Std	The match standard for a maturity postname. For example, Sr. is a match standard or alias for Senior. If the cleansing package does not have an alias entry, the output field is empty.
Name_Connector	None	The connector component of a dual name. For example, and.

Generated field name	Content type	Description
Name_Designator	None	Name designator such as Attn: or c/o.
Name_Special	None	Term that generically describes a person. For example, occupant or current resident.
North_American _Phone	Phone	An entire North American Numbering Plan (NANP) phone number.
North_American _Phone_Area_Code	Phone	The area code parsed from the phone number.
North_American _Phone_Extension	Phone	An extension parsed from the phone number.
North_American _Phone_Line	Phone	The last four numbers (excluding an extension) parsed from a phone number. In (123) 456-7890, 7890 is returned.
North_American _Phone_Prefix	Phone	The middle three numbers parsed from a phone number. In (123) 456-7890, 456 is returned.
North_American _Phone_Type	Phone	The type of phone number that was parsed, if it is included with the input. For example, Home or Work.
Person	None	Set of components that define a single person. For example, Thomas Williams-Doyle Sr., M.D.
Phone	Phone	Shows the phone number that was identified as either North American or International.
Prename	Prename	Prename (for example, Mr.).
Prename_Match_Std1-6	Prename_Match_Std	The match standard for a prename. For example, Mr. is the match standard or alias for Mister. If the cleansing package does not have an alias entry, the output field is empty. Compare with Match_Prename.
Rule_Label	None	Retrieves the rule that parsed the indicated item.
Score	None	Retrieves the confidence score for a parsed item.
SSN	SSN	The entire Social Security number.
SSN_Area	SSN	The first three numbers of the Social Security number.
SSN_Group	SSN	The fourth and fifth numbers within a Social Security number.

Generated field name	Content type	Description
SSN_Serial	SSN	The last four numbers in a Social Security number.
Status_Code	None	The code describing how the data was standardized when the job is set for standardization, or how the data could be standardized when the job settings are not set for standardization.
Title	Title	Job or occupational title of a person. For example, Manager.
Title_Match_Std1-6	Title_Match_Std	The match standard for title. For example, CFO is the match standard or alias for Chief Financial Officer. If the dictionary does not have an alias entry, the output field is empty.
UDPM	None	Attribute field defined in User-defined pattern rules in Cleansing Package Builder Reference Data.
UDPM_ Subcomponent1-5	None	Subcomponents of the UDPM attribute field defined in a User-defined pattern rule.

3.5.4.8 DSF2® Walk Sequencer

To add walk sequence information to your data, include the DSF2 Walk Sequencer transform in your data flow. You can then send your data through presorting software to qualify for the following walk-sequence discounts:

- Carrier Route
- Walk Sequence
- 90% Residential Saturation
- 75% Total Active Saturation

DSF2 walk sequencing is often called "pseudo" sequencing because it mimics USPS walk sequencing. Where USPS walk-sequence numbers cover every address, DSF2 walk sequence processing provides "pseudo" sequence numbers for the addresses only in that particular file.

The software uses DSF2 data to assign sequence numbers for all addresses that are DPV-confirmed delivery points (DPV_Status contains “Y”, “S”, or “D”).

Other addresses present in your output file that are not valid DPV-confirmed delivery points will be blank or contain "0000" in the Walk_Sequence_Number output field. For example, if addresses have a DPV_Status of “N”, the walk sequence number is “0000”. If DPV_Status is blank, the walk sequence number field is blank.

i Note

Before processing your data with the DSF2 Walk Sequencer transform, you must first process it through CASS-certified software. This can be accomplished by processing data first with the USA Regulatory Address Cleanse transform with CASS certification enabled. The output from that processing can then be used as input for the DSF2 Walk Sequencer transform.

To help set up Data Services jobs, you can use Data Quality blueprints and other content objects, including several blueprints to run DSF2 certifications. These blueprints are located in **<LINK_DIR>**`\DataQuality\Certifications`.

3.5.4.8.1 Common

Option	Description
Run as Separate Process	*Yes*: Splits the transform into a separate process. *No*: Keeps the transform in the same process as the rest of the data flow.

3.5.4.8.2 Reference Files

Enter the path to the Delivery Statistics directory file (Delstats). It is best to use the substitution variable $ $RefFilesAddressCleanse for the Option Value. The substitution parameter represents the path, and you can change it dynamically.

ption	Description
Delstats Directory	`dsf.dir` Type the path and file name of the delivery statistics directory, or use the substitution variable $ $RefFilesAddressCleanse. SAP provides this file with the U.S. National Directory delivery. The Delivery Statistics directory file provides counts of business and residential addresses per Postcode1(ZIP Code), per Sortcode route (carrier route).

3.5.4.8.3 Processing Options

Option	Description
Site Location	Indicates at which site the current job was processed (if your company has multiple locations that provide DSF2 walk sequence processing). Enter the name of your company's location where the DSF2 walk sequence processing is performed for this job. Use the substitution variable $$DSF2SiteLocation if the site location doesn't change often.
USPS Certification Testing Mode	Specifies the type of certification you are performing for DSF2 walk sequence if applicable: *None:* Not performing any DSF2 walk sequence certifications (this is the default setting). Choose this option if you are processing a job that will not be submitted to the USPS for DSF2 certification. *Invoice:* Performing certification for DSF2 walk sequence invoice certification. For invoice certification, you are certifying that the software assigns walk sequence numbers correctly, and creates the Delivery Sequence Invoice report. *Sequence:* Performing certification for DSF2 sequence certification. For sequence certification, you are certifying that the software assigns walk sequence numbers correctly, creates the Delivery Sequence Invoice report, and creates the SEQ log file.

3.5.4.8.4 USPS License Information

The USPS License Information options in the DSF2 Walk Sequencer transform are required:

Option	Description
DSF2 License ID	Enter your DSF2 identification number, as the USPS assigned it to you. You can use the substitution variable $ $DSF2LicenseeID.
Licensee Name	Enter the name of the DSF2 licensed service provider. You can use the substitution variable $$USPSLicenseeName.
List ID	Enter the unique 6-digit identification code that you (the DSF2 licensee) assigned to the customer who owns the list.

3.5.4.8.5 Data Collection Config

The settings in this group of options control the break key formation. The break key optimizes your data flow by sorting your data to form collections of input records that have the same Postcode1 and Sortcode_Route field values.

Option	Description
Replace Null With Space	Specifies whether to convert NULL values to blank spaces in the break key fields. Eliminating NULL values helps to standardize data in the field so that break groups are formed properly and are consistent in size. *Yes*: Convert NULL to blank spaces. *No*: Do not convert NULL to blank spaces.
Right Pad With Spaces	Specifies whether to right pad the break key field with spaces. Because the break key is used for sorting and aggregating, it is sensitive to the position in which data is placed. By right-padding the break key fields you can help ensure that break groups are formed properly and are consistent in size. *Yes*: Right-pad fields with blank spaces. *No*: Do not right-pad fields with blank spaces. ➜ Tip If the *Replace NULL with space* option is set to *Yes* and the *Right pad with spaces* option is set to *Yes*, then fields with NULL values will be replaced with all spaces on the right (to the length of the field).
Presorted Data	Specifies whether the input data has been presorted or not. To make your input data more consistent, it is best to have the software sort data by the break key fields (Postcode1 and Sortcode_Route). *Yes*: The input data has already been presorted by Postcode1 and Sortcode_Route. *No*: The input data has not been sorted yet. ➜ Tip Choosing *No* allows the software to sort your data by Postcode1 and Sortcode_Route. This is the preferred setting for this option.

3.5.4.8.6 Input fields

Field	Description
Delivery_Point	The two-digit DPBC code.
DPV_Status	The DPV status component that is generated for this record. *D:* The primary range is a confirmed delivery point, but the secondary range was not available on input. *L:* The address triggered DPV locking. *N:* The address is not a valid delivery point. The Walk_Sequence_Number output field is 0000. *S:* The primary range is a valid delivery point, but the parsed secondary range is not valid in the DPV directory. *Y:* The address is a confirmed delivery point. The primary range and secondary range (if present) are valid. **`<blank>`**: A blank output value indicates that Enable DPV is set to No, DPV processing is currently locked, or the transform cannot assign the input address. The Walk_Sequence_Number output field is blank.
DSF2_Business_Indicator (optional)	Residential/business indicator. You may use this information to lower your parcel-shipping costs. (Some parcel delivery services charge more for delivery to residential addresses.) *Y:* Business address. *N:* Not a business address. **`<blank>`**: Address was not looked up.
LOT	The Line-of-Travel number.
LOT_Order	The Line-of-Travel sortation: *A:* Ascending *D:* Descending LOT codes are required for non-automated, CART presorting in Standard Mail, Enhanced Carrier Route Subclass.
Postcode1	The five-digit primary ZIP Code. It does not include the four digit ZIP4 Code.
Postcode2	The four-digit ZIP4 code. On a mail piece, this code follows the primary postal code with a hyphen placed between, for example, 54601-1234.
Sortcode_Route	The four-digit carrier route number.

3.5.4.8.7 Output fields

The software outputs walk-sequence number information to the fields listed in the table below.

Field	Description
Active_Del_Discount	Indicates whether the postcode1/sortcode route combination qualifies for the 75% total active deliveries discount. *Y:* The postcode1/sortcode route combination qualifies for the 75% total active deliveries discount. *N:* The postcode1/sortcode route combination does not qualify for the 75% total active deliveries discount. `<blank>`: The record was not sequenced. Tip Active deliveries include residential, business, and PO Box addresses.
Residential_Sat_Discount	Indicates whether the postcode1/sortcode route combination qualifies for the 90% residential saturation discount. *Y:* The postcode1/sortcode route combination qualifies for the 90% residential saturation discount. *N:* The postcode1/sortcode route combination does not qualify for the 90% residential saturation discount. `<blank>`: The record was not sequenced.
Sortcode_Route_Discount	Indicates whether the postcode1/sortcode route combination qualifies for the Sortcode (Carrier Route) discount. *Y:* The postcode1/sortcode route combination qualifies for the Sortcode discount. *N:* The postcode1/sortcode route combination does not qualify for the Sortcode discount. `<blank>`: The record was not sequenced. Tip Mailers must have 10 or more deliveries to the same postcode1/ sortcode combination to qualify for the discount.
Walk_Sequence_Discount	Indicates whether the postcode1/sortcode route combination qualifies for the walk sequence discount.

Field	Description
	Y: The postcode1/sortcode route combination qualifies for the walk sequence discount *N:* The postcode1/sortcode route combination does not qualify for the walk sequence discount. `<blank>`: The record was not sequenced. ➜ Tip Mailers must have 125 or more sequenced delivery points for each postcode1/sortcode route combination to qualify for the discount.
Walk_Sequence_Number	Indicates the sequence number from 0000 to 9999. i Note If Postcode2 field is blank, this field is blank. i Note If Postcode2 field is not blank and *DPV_Status* is *N*, then the field contains 0000.

3.5.4.9 Geocoder

How the Geocoder transform works

The Geocoder transform uses geographic coordinates expressed as latitude and longitude, addresses, and point-of-interest (POI) data. Using the transform, you can append addresses, latitude and longitude, census data (US only), and other information to your data.

Based on mapped input fields, the Geocoder transform has three modes of geocode processing:

- *Address geocoding*: The Geocoder transform assigns geographic data. Based on the completeness of the input address data, the Geocoder transform can return multiple levels of latitude and longitude data. Appending different levels of latitude and longitude information to your data may help your organization to target certain population sizes and other regional geographical data.
- *Reverse geocoding*: The Geocoder transform identifies the closest address or point of interest based on an input reference location

- *POI textual search*: The Geocoder transform uses address fields and POI name or type fields as search criteria to match with points of interest. The results are output in the Result_List XML output field.

Typically, the Geocoder transform is used in conjunction with the Global Address Cleanse or USA Regulatory Address Cleanse transform.

Related Information

Designer Guide: Data Quality, Address geocoding [page 490]
Designer Guide: Data Quality, Reverse geocoding [page 495]
Designer Guide: Data Quality, POI textual search [page 503]

3.5.4.9.1 Content objects

Transform configurations

A transform configuration is a transform with preconfigured input fields, output fields, and options that can be used in multiple data flows. These are useful if you repeatedly use a transform with specific options and input and output fields.

When Data Services is installed, read-only transform configurations are provided for the Data Quality transforms.

You can use transform configurations in your data flows or as an example of a typical transform. After you place an instance of the transform configuration in a data flow, you can override these preset defaults. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one.

Sample blueprints and other objects

We have created Data Quality blueprints and other content objects to help you set up Data Services jobs. We've identified a number of common scenarios that you are likely to perform with Data Services. For each scenario, we've included a blueprint that is already set up to solve the business problem in that scenario.

Related Information

Downloading blueprints and other content objects [page 1128]
Transform configurations [page 1123]

3.5.4.9.2 Geocoder options

The Geocoder transform includes options that control how geocoding data is appended to your data.

3.5.4.9.2.1 Common options

Option	Description
Run As Separate Process	*Yes:* Splits the transform into a separate process. *No:* Keeps the transform in the same process as the rest of the data flow.

3.5.4.9.2.2 Report and analysis

Use this option to generate report data for the Geocoder transform.

Option	Description
Generate Report Data	Specifies whether to generate report data for this transform. *Yes:* Generates report data for this transform. *No:* Turns off report data generation. If you do not need to generate reports (during testing, for example), you should set this option to *No* to improve performance. i Note This option is only available for the address geocoding mode. In the reverse geocoding and POI textual search modes, if you set the value to Yes, you will get a warning and no report is generated.

Related Information

Management Console Guide: Geocoder Summary report [page 2021]

3.5.4.9.2.3 Geocoder options

Specifies the assignment levels and other options. This option group is required.

All countries are supported.

Supported directory formats:

- *Basic*: Supports the address geocoding mode. It returns centroid-level and address-level latitude and longitude information.
- *Advanced*: Supports the address geocoding, reverse geocoding, and POI textual search modes. It returns range-based and centroid-level latitude and longitude information and point-of-interest information.

Option	Description	Supported directory format
Best Assignment Level	Specifies the depth of assignment for the latitude and longitude output fields. This option is used for address geocoding mode, and also for reverse geocoding with address mode. *Preferred:* Assigns to the finest depth. By default, this assigns to the primary number level. *Primary Number:* Assigns to the primary number level. *Postcode:* Assigns to the postcode level. *Locality:* Assigns to the locality, city, or suburb level. *Smallest Area:* The software first attempts to assign to the primary number. If the primary number is not returned, then it assigns based on postcode or locality, depending on which level is the smaller area. The Geocoder transforms compares the locality data to the postcode data, and then assigns to the level based on the smallest area. For example, the French postcode 75014 is a smaller area than the locality of Paris. Note For the Preferred, Primary Number, and Smallest Area values, if the Locality Assignment Threshold or Postcode Assignment Threshold option is not set to None, it will do locality or postcode centroid assignment.	Basic, Advanced
Distance Unit	Specifies the unit of distance used for the radius. *Kilometers* *Miles* The value of the Option_Distance_Unit input field takes precedence over the value of the Distance Unit option. The value of the Distance Unit option is only used if the Option_Distance_Unit input field is not mapped or is invalid, blank, or NULL.	Advanced
Locality Assignment Threshold	Limits the level of locality centroid assignment. For example, setting the option to Locality1 excludes Locality2–4 during assignment, even though the software may return values at those levels. However, if you set the option to Locality4 and there is no Locality4 data, the finest available data is returned, even if that is a Locality2 level.	Basic, Advanced

<table>
<tr><th>Option</th><th>Description</th><th>Supported directory format</th></tr>
<tr><td></td><td>This option is used for address geocoding mode, and also for reverse geocoding with address mode.

Locality1–4: Returns the locality level that you choose. Locality1 is the most general and Locality4 is the most specific.

<table><tr><th>Address</th><th>Locality level</th></tr><tr><td>Church Cottage</td><td>Locality3</td></tr><tr><td>Pemborough</td><td>Locality2</td></tr><tr><td>Bristol</td><td>Locality1</td></tr></table>
In this example, there is no Locality4. If you choose Locality4, the finest depth, Locality3, is returned.

None: Skips the specific assignment level. Use this setting if you do not want to return an assignment threshold on locality.

Preferred: Assigns to the finest depth at the locality level.</td><td></td></tr>
<tr><td>Max Records</td><td>Specifies the maximum number of records that can be returned. You can enter a number up to 100.

The value of the Option_Max_Records input field takes precedence over the value of the Max Records option. The value of the Max Records option is only used if the Option_Max_Records input field is not mapped or is blank.</td><td>Advanced</td></tr>
<tr><td>Offset Coordinates</td><td>Specifies whether the offset values of latitude and longitude are returned when the side of the street is known. This option is used for address geocoding mode, and also for reverse geocoding with address mode.

Yes: Returns the offset values.

No: Returns the center value regardless of whether the side of the street is known.</td><td>Advanced</td></tr>
<tr><td>Postcode Assignment Threshold</td><td>Limits the level of postcode centroid assignment. For example, setting the option to Postcode1 excludes the other levels during assignment, even though the application may return values at those levels.

This option is used for address geocoding mode, and also for reverse geocoding with address mode.

Postcode Full: Assigns to the entire extended postcode. For example, in the USA, it assigns to the 5-digit postcode and all four digits of the ZIP+4.

Postcode1: Assigns to the city or postcode area. For example, in the USA, it assigns to the 5-digit ZIP Code.</td><td>Basic, Advanced</td></tr>
</table>

Option	Description	Supported directory format
	Postcode2 Partial: Assigns to the first few characters of the extended postcode. For example, in the USA, it assigns the 5-digit postcode and the first two digits of the ZIP+4. *Preferred:* Assigns to the finest depth at the postcode level. *None:* Skips the specific assignment level. Use this setting if you do not want to return an assignment threshold on postcode.	
Radius	The distance from a specified reference point used to identify an area in which matching records are located. The value of the Option_Radius input field takes precedence over the value of the Radius option. The value of the Radius option is only used if the Option_Radius input field is not mapped or is blank. If a radius is not specified, a default radius of one kilometer is used. In reverse geocoding mode, the maximum radius distance is 111 kilometers or 68.97 miles.	Advanced

3.5.4.9.3 Reference files

Reference files are directory paths required by the Geocoder transform to process your data.

You can use a substitution variable for the directory location. The substitution variable for reference files is $ $RefFilesGeocoder. Use this substitution variable if you installed the directories in the path relative to where Data Services is installed; for example, **<INSTALL_DIR>**`\Data Services\DataQuality\reference_data`. If you installed to a different location, you can change the substitution variable while designing the work flow. For more information about variables, see the Variables and Parameters section in the *Data Services Designer Guide.*

You can duplicate this path by right-clicking and selecting *Duplicate option* to point to additional directory paths. However, you cannot have the same directory file located in both directories. In the *Directory Path* option, only specify the path. Do not specify the directory file names.

For information about downloading directories, see the latest directories update.

3.5.4.9.4 Geocoder fields

The Geocoder transform requires that you map fields on input and output. These mappings inform the transform how to process the data in the field.

Related Information

Input fields [page 1184]

Output fields [page 1186]
Result_List XML output fields [page 1192]
Information codes [page 1194]

3.5.4.9.4.1 Input fields

The following are recognized input fields that you can use in the input mapping for the Geocoder transform. The fields are listed alphabetically.

All countries are supported.

The table also shows the input field availability based on currently supported directory formats. Supported directory formats:

- *Basic*: Supports the address geocoding mode. It returns centroid-level and address-level latitude and longitude information.
- *Advanced*: Supports the address geocoding, reverse geocoding, and POI textual search modes. It returns range-based and centroid-level latitude and longitude information and point-of-interest information.

Input field name	Category	Description	Supported directory format
Country	Address	The two-character ISO country code.	Basic, Advanced
Latitude	Latitude/ Longitude	A relative distance north or south of the equator, measured in 0-90 degrees.	Advanced
Locality1–4	Address	The city, town, or suburb and any additional related information.	Basic, Advanced
Longitude	Latitude/ Longitude	A relative distance east or west of the Greenwich meridian, measured in 0-180 degrees.	Advanced
Option_Distance_Unit	Search Filter	The unit of distance used for the radius. Valid values are: *Kilometers* *Miles* The value of the Option_Distance_Unit input field takes precedence over the value of the Distance Unit option. The value of the Distance Unit option is only used if the Option_Distance_Unit input field is not mapped or is invalid, blank, or NULL. This setting is dynamic. If you change this setting, you do not have to terminate and reinitialize the transform in order for the new configuration to be recognized.	Advanced
Option_Max_Records	Max Records	The maximum number of records that can be returned. You can enter a number up to 100.	Advanced

Input field name	Category	Description	Supported directory format
		A value greater than 0 outputs multiple results as XML to the Result_List output field rather than to individual output fields. The value of the Option_Max_Records input field takes precedence over the value of the Max Records option. The value of the Max Records option is only used if the Option_Max_Records input field is not mapped or is blank or NULL. This setting is dynamic. If you change this setting, you do not have to terminate and reinitialize the transform in order for the new configuration to be recognized.	
Option_Radius	Search Filter	The distance from a specified reference point used to identify an area in which matching records are located. The value of the Option_Radius input field takes precedence over the value of the Radius option. The value of the Radius option is used only when the Option_Radius input field is not mapped or is blank or NULL. This setting is dynamic. If you change this setting, you do not have to terminate and reinitialize the transform in order for the new configuration to be recognized. In reverse geocoding mode, the maximum radius distance is 111 kilometers or 68.97 miles.	Advanced
POI_Name	Address POI	The name of a point of interest, such as the Washington Monument.	Advanced
POI_Type	Address POI	The point-of-interest type expressed as a number; for example, for one vendor 5999 is historical monument. In POI textual search mode, to return multiple point-of-interest types, concatenate POI type codes using a colon as a delimiter. For example, to return all schools (type 8211) and libraries (type 8231) within a defined area, you would enter: `8211:8231` The POI types and their corresponding codes differ depending on the data vendor that you use. For a detailed list of available POI types, see the vendor-specific directory update.	Advanced

Input field name	Category	Description	Supported directory format
Postcode1–2	Address	The postal code and a secondary postal code, if available.	Basic, Advanced
Primary_Name1–4	Address	The street name.	Basic, Advanced
Primary_Number	Address	The premise number.	Basic, Advanced
Primary_Postfix1	Address	Abbreviated directional (N, S, NW, SE) that follows a street name.	Basic, Advanced
Primary_Prefix1	Address	Abbreviated directional (N, S, NW, SE) that precedes a street name.	Basic, Advanced
Primary_Type1–4	Address	Abbreviated type of primary name (St., Ave., or Pl.).	Basic, Advanced
Region1–2	Address	The region symbol of the state, province, or territory.	Basic, Advanced
Search_Filter_Name	Search Filter	Search criteria for a point-of-interest name.	Advanced

Search_Filter_Type	Search Filter	Search criteria for a point-of-interest type, expressed as a four-digit number; for example, for one vendor 5999 is historical monument.
		If you want to return an address only, enter ADDR.
		To return multiple point-of-interest types, concatenate POI type codes using a colon as a delimiter. For example, to return all schools (type 8211) and libraries (type 8231) within a defined area, you would enter:
		`8211:8231`
		To return a point-of-interest type and its address, enter:
		`5999:ADDR`
		The POI types and their corresponding codes differ depending on the data vendor that you use. For a detailed list of available POI types, see the vendor-specific directory update.
			Advanced

3.5.4.9.4.2 Output fields

The following are recognized output fields that you can use in the output mapping for the Geocoder transform. The fields are listed alphabetically. The table also shows the output field availability based on currently supported countries and directory formats.

Supported countries:

- *All*: All countries
- *USA*

Supported directory formats:

- *Basic*: Supports the address geocoding mode. It returns centroid-level and address-level latitude and longitude information.
- *Advanced*: Supports the address geocoding, reverse geocoding, and POI textual search modes. It returns range-based and centroid-level latitude and longitude information and point-of-interest information.

Output field name	Category	Description	Supported countries	Supported directory format
Address_Line	Address	A line of data in an address that contains the primary address. The primary address can contain components such as the primary range, primary name, directionals (post- and pre-), and suffix.	All	Basic, Advanced
Assignment_Level	Assignment Level	The level to which the transform matches the address to the data in the reference fields (directories). *PRE:* Primary Range Exact assigns to the exact location of the address; for example, 123 Main St. This is the most precise level of assignment. To obtain the PRE, you must map either the POI_Type input field or the Primary_Name and Primary_Number input fields. *PRI:* Primary Range Interpolated assigns to the level of the address range; for example, 100-500 Main St. *L1–4:* Locality1–4 assigns to the level of city, town, or suburb. *P1:* Postcode1 assigns to the level of Postcode1. *P2P:* Postcode2 Partial assigns the full Postcode1 and the first few characters of Postcode2. *PF:* Postcode Full assigns to the level of Postcode1 and Postcode2, when available.	All	Basic, Advanced
Assignment_Level_Locality	Assignment Level	The level to which the transform assigns the locality.	All	Basic, Advanced

Output field name	Category	Description	Supported countries	Supported directory format
		L1–4: Returns up to four locality levels. L1 is the most general and L4 is the most specific.		
Assignment_Level_Postcode	Assignment Level	The level to which the transform assigns the postcode. *P1:* Postcode1 assigns to the level of Postcode1. *P2P:* Postcode2 Partial assigns the full Postcode1 and the first few characters of Postcode2. *PF:* Postcode Full assigns to the level of Postcode1 and Postcode2, when available.	All	Basic, Advanced
Census_Tract_Block	Census	The census tract code as defined by the government for reporting census information. Census tracts are small, relatively permanent statistical subdivisions of a county.	USA	Basic, Advanced
Census_Tract_Block_Prev	Census	The census tract code in the previous version of census data.	n/a	n/a
Census_Tract_Block_Group	Census	The census tract block group code as defined by the government for reporting census information. These codes are used for matching with demographic-coding databases. In the USA, the first six digits contain the tract number (for example, 002689); the next digit contains the block group (BG) number within the tract, and the last three digits contain the block code. The BG is a cluster of census blocks that have the same first digit within a census tract. For example, BG 6 includes all blocks numbered from 6000 to 6999.	USA	Basic, Advanced
Census_Tract_Block_Group_Prev	Census	The census tract block group code in the previous version of census data.	n/a	n/a
Country_Code	Address	The two-character ISO country code.	All	Basic, Advanced
Distance	Distance	The distance from the input address, geographical coordinates, or point of	All	Advanced

Output field name	Category	Description	Supported countries	Supported directory format
		interest to the closest address or point of interest.		
Gov_County_Code	Census	A unique county code as defined by the government for reporting census information. For example, in the USA, this is a Federal Information Processing Standard (FIPS) three-digit county code.	USA	Basic, Advanced
Gov_Locality1_Code	Census	A unique code for an incorporated municipality such as a city, town, or locality as defined by the government for reporting census information.	USA	Basic
Gov_Region1_Code	Census	A unique region code as defined by the government for reporting census information. For example, in the USA, this is a Federal Information Processing Standard (FIPS) two-digit state code.	USA	Basic, Advanced
Info_Code	Info Code	A three-character code that provides information about the geocoding results. The status for address and point-of-interest geocoding assignment is indicated in the third character. The status for reverse geocoding assignment is indicated in the second and third characters. If assigned to the best level, the Info_Code field is blank. The first character is not used at this time. For more information, see *Information codes* [page 1194].	All	Basic, Advanced
Latitude	Latitude/ Longitude	The latitude at the best assigned level (0–90 degrees north or south of the equator) in the format 45.32861.	All	Basic, Advanced
Latitude_Locality	Latitude/ Longitude	The latitude at the locality level centroid of the city, town, locality, or suburb in the format 45.32861.	All	Basic, Advanced
Latitude_Postcode	Latitude/ Longitude	The latitude at the postcode level centroid of the postcode in the format 45.32861.	All	Basic, Advanced

Output field name	Category	Description	Supported countries	Supported directory format
Latitude_Primary_Number	Latitude/ Longitude	The latitude at the primary number level centroid of the primary number in the format 45.32861.	All	Basic, Advanced
Locality1–4	Address	The city, town, or suburb and any additional related information.	All	Basic, Advanced
Longitude	Latitude/ Longitude	The longitude at the best assigned level (0–180 degrees east or west of Greenwich meridian) in the format 123.45833.	All	Basic, Advanced
Longitude_Locality	Latitude/ Longitude	The longitude at the locality level centroid of the city, town, locality, or suburb in the format 123.45833.	All	Basic, Advanced
Longitude_Postcode	Latitude/ Longitude	The longitude at the postcode level centroid of the postcode in the format 123.45833.	All	Basic, Advanced
Longitude_Primary_Number	Latitude/ Longitude	The longitude at the primary number level centroid of the primary number in the format 123.45833.	All	Basic, Advanced
Metro_ Stat_Area_Code	Census	The metropolitan statistical area. For example, in the USA, the 0000 code indicates the address does not lie in a metropolitan statistical area; usually a rural area. A metropolitan statistical area has a large population that has a high degree of social and economic integration with the core of the area. The area is defined by the government for reporting census information.	USA	Basic
Metro_ Stat_Area_Code_Prev	Census	The metropolitan statistical area in the previous version of census data.	n/a	n/a
Minor_Div_Code	Census	The minor civil division or census county division code when the minor civil division is not available. The minor civil division designates the primary government and/or administrative divisions of a county such as a civil township or precinct. Census county division are defined in a state or province that does not have a well-defined minor civil division. The area	USA	Basic, Advanced

Output field name	Category	Description	Supported countries	Supported directory format
		is defined by the government for reporting census information.		
Minor_Div_Code_Prev	Census	The minor civil division or census county division code in the previous version of census data.	n/a	n/a
POI_Name	Address POI	The point of interest name, such as the Washington Monument.	All	Advanced
POI_Type	Address POI	The point of interest type expressed as a four-digit number; for example, 5999 (historical monument).	All	Advanced
Population_Class_Locality1	Population	Indicates that the population falls within a certain size. *0:* Undefined. The population may be too large or small to provide accurate data. *1:* Over 1 million. *2:* 500,000 to 999,9999. *3:* 100,000 to 499,999. *4:* 50,000 to 99,999. *5:* 10,000 to 49,999. *6:* Less than 10,000.	All	Advanced
Postcode	Address	The postal code.	All	Basic, Advanced
Postcode1–2	Address	The postal code and a secondary postal code, if available.	All	Basic, Advanced
Primary_Name1–4	Address	The street name.	All	Basic, Advanced
Primary_Number	Address	The premise number.	All	Basic, Advanced
Primary_Postfix1	Address	Abbreviated directional (N, S, NW, SE) that follows a street name.	All	Basic, Advanced
Primary_Prefix1	Address	Abbreviated directional (N, S, NW, SE) that precedes a street name.	All	Basic, Advanced
Primary_Range_High	Address	The high value of a primary number range.	All	Basic, Advanced
Primary_Range_Low	Address	The low value of a primary number range.	All	Basic, Advanced

Output field name	Category	Description	Supported countries	Supported directory format
Primary_Type1–4	Address	Abbreviated type of primary name (St., Ave., or Pl.).	All	Basic, Advanced
Region1–2	Address	The region symbol of the state, province, or territory.	All	Basic, Advanced
Result_List	Results	The XML output when multiple records are returned for a search.	All	Advanced
Result_List_Count	Results	The number of results in the Result_List output field.	All	Advanced
Side_Of_Primary_Address	Side of Street	Indicates that the location is on the L (left) or R (right) side of the street when moving north, northeast, northwest, or east.	All	Advanced
Stat_Area_Code	Census	A core-based statistical area code where an area has a high degree of social and economic integration within the core that the area surrounds. The area is defined by the government for reporting census information.	USA	Basic
Stat_Area_Code_Prev	Census	The statistical area code in the previous version of census data.	n/a	n/a

Related Information

Information codes [page 1194]

3.5.4.9.4.3 Result_List XML output fields

The following are recognized output fields that you can use in the Result_List XML output field in the Geocoder transform. The fields are listed alphabetically.

Output field name	Category	Description
Address_Line	Address	A line of data in an address that contains the primary address. The primary address can contain components such as the primary range, primary name, directionals (post- and pre-), and suffix.
Assignment_Level	Assignment Level	The level to which this transform matched the address to the data in the reference fields (directories).

Output field name	Category	Description
		PRE: Primary Range Exact assigns to the exact location of the address; for example, 123 Main St. This is the most precise level of assignment. To obtain the PRE, you must map either the POI_Type input field or the Primary_Name and Primary_Number input fields. *PRI:* Primary Range Interpolated assigns to the level of the address range; for example, 100-500 Main St. *L1–4:* Locality1-4 assigns to the level of city, town, or suburb. *P1:* Postcode1 assigns to the level of Postcode1. *P2P:* Postcode2 Partial assigns the full Postcode1 and the first few characters of Postcode2. *PF:* Postcode Full assigns to the level of Postcode1 and Postcode2, when available.
Country_Code	Address	The two-character ISO country code.
Distance	Distance	The distance from the input address, geographical coordinates, or point of interest to the closest address or point of interest.
Latitude	Latitude/Longitude	The latitude at the best assigned level (0–90 degrees north or south of the equator) in the format 45.32861.
Locality1–4	Address	The city, town, or suburb and any additional related information.
Longitude	Latitude/Longitude	The longitude at the best assigned level (0–180 degrees east or west of Greenwich meridian) in the format 123.45833.
POI_Name	Address POI	The point of interest name, such as the Washington Monument.
POI_Type	Address POI	The point of interest type expressed as a four-digit number; for example, 5999 (historical monument).
Postcode	Address	The postal code.
Postcode1–2	Address	The postal code and a secondary postal code, if available.
Primary_Name1–4	Address	The street name.
Primary_Number	Address	The premise number.
Primary_Postfix1	Address	Abbreviated directional (N, S, NW, SE) that follows a street name.
Primary_Prefix1	Address	Abbreviated directional (N, S, NW, SE) that precedes a street name.

Output field name	Category	Description
Primary_Range_High	Address	The high value of a primary number range.
Primary_Range_Low	Address	The low value of a primary number range.
Primary_Type1–4	Address	Abbreviated type of primary name (St., Ave., or Pl.).
Ranking	Ranking	A numeric value that indicates how well the returned records match the input field based on the match score. A record with a ranking of 1 has the highest match score.
Region1–2	Address	The region symbol of the state, province, or territory.

The following is an example for a result list that has one record:

```
<RESULT_LIST>
  <RECORD>
    <ADDRESS_LINE>332 FRONT ST</ADDRESS_LINE>
    <ASSIGNMENT_LEVEL>PRE</ASSIGNMENT_LEVEL>
    <COUNTRY_CODE>US</COUNTRY_CODE>
    <DISTANCE>0.3340</DISTANCE>
    <LATITUDE>43.811616</LATITUDE>
    <LOCALITY1>LA CROSSE</LOCALITY1>
    <LONGITUDE>-91.256695</LONGITUDE>
    <POI_NAME>ABC COMPANY</POI_NAME>
    <POI_TYPE>5800</POI_TYPE>
    <POSTCODE>56001-4023</POSTCODE>
    <POSTCODE1>56001</POSTCODE1>
    <POSTCODE2>4023</POSTCODE2>
    <PRIMARY_NAME1>FRONT</PRIMARY_NAME1>
    <PRIMARY_NUMBER>332</PRIMARY_NUMBER>
    <PRIMARY_TYPE1>ST</PRIMARY_TYPE1>
    <RANKING>1</RANKING>
    <REGION1>WI</REGION1>
  </RECORD>
</RESULT_LIST>
```

3.5.4.9.4.4 Information codes

The Info_Code output field is a three-character code that provides information about geocoding results.

The status for the address geocoding and POI textual search modes is indicated in the third character. The status for reverse geocoding is indicated in the second and third characters. If assigned to the best level, the Info_Code field is blank. The first character is not used at this time.

Use the following table to determine the code assigned to the Info_Code output field.

Information code	Description
1	Reference data is not available for the input country. Verify that the directory is installed and the reference path to the directory is valid.
2	Address-level reference data is not available for the input data. When Best Assignment Level is set to Primary Number and the address directory is unavailable or doesn't exist, this code is output.

Information code	Description
	Verify that the directory is installed and the reference path to the directory is valid.
3	Centroid-level reference data is not available for the input data. When the Best Assignment Level is set to Locality or Postcode, and the address directory is unavailable or doesn't exist, this code is output. Verify that the directory is installed and the reference path to the directory is valid.
4	Assignment is limited. The input data is insufficient or incorrect to match the reference data. When the Best Assignment Level fails, this code is output. The assignment may be made to a lower assignment level than the one specified. For example, if you set Best Assignment Level to Primary Number and the Primary Number field is blank, the assignment may be at the postcode or locality level, if the data is available. Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.
5	The input data does not match anything in the reference data. When the input record does not match the directory data for the Best Assignment Level or a lower assignment level, this code is output. Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.
6	The input data assigns ambiguously in the reference data. There is a tie for the Best Assignment Level. The input record matches several records in the directory data and the software cannot decide which one is the best. For example, if the reference data has two records: Record 1: 100 Main St La Crosse WI 54650 Record 2: 100 Main St Bt Micts WI 54650 When input with 100 Main ST WI 54650 without a locality name, the 006 information code for ambiguous assignment is output. Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.
7	The input data is blank or invalid. For example, if the US Postcode1 is a five-digit string and your input data is a six-digit string, the 007 information code is output. Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.
8	The input data is insufficient. When the input data for the selected Best Assignment Level is blank, this code is output. For example, this code is output when you set the Best Assignment Level to Primary Number and the input data is blank for Primary Number.
9	The POI type provided on input is invalid. The point-of-interest type is not correct.

Information code	Description
	Verify your input POI type with the POI types described in the appropriate directory update letter.
A	The POI input data was not used. The POI name or type does not match the directory data. A PRI or PRE level assignment was made; however, the input POI name and POI type were not used for the assignment.
B	The input data was not found. This code only occurs when an address is input during reverse geocoding assignment. The input address doesn't match the directory data and reverse geocoding cannot be performed based on the address.
C	Not all results returned for the input data, because the number of results exceeds the specified Max_Record. This code only occurs during reverse geocoding assignment. Increase the Max_Record value.
D	Not all results were returned for the input data, because the results exceed the field length available in the Result_List XML output field. This code only occurs during reverse geocoding assignment. The Geocoder transform allows a 60000 field length in the Result_List field.
E	The closest latitude and longitude to the input address is returned because the input house number does not exist in the geocoder directories. For example: Input address: 100 Main St La Crosse WI 54650 Directory data: 1-88 Main St La Crosse WI 54650 The software returns the latitude and longitude values for 88 Main St with the 00E information code to indicate that the house number does not match the directory data.

Related Information

Output fields [page 1186]

3.5.4.9.5 Directories

The Geocoder transform is flexible enough to accept new country directory data immediately after the directory data is released. There is no need to wait for the next SAP Data Services release to begin using new country directory data.

The Geocoder directories are designed specifically for use with the Geocoder transform. You must install the directories and point to them in the Reference Path. Your system administrator should have already installed these files to the appropriate locations.

If you start with the sample transform configuration, the reference path is completed with a substitution variable called $$RefFilesGeocoder. By default, this variable points to the reference data folder of the Data Services directory. You can change the location by editing the substitution variable. For more information about variables, see the “Variables and Parameters” section in the *Data Services Designer Guide.*

SAP offers two directory formats:

- *Basic*: Supports the address geocoding mode. It returns centroid-level and address-level latitude and longitude information.
- *Advanced*: Supports the address geocoding, reverse geocoding, and POI textual search modes. It returns a range-based and centroid-level latitude and longitude information and point-of-interest information

Directory updates

A quarterly vendor-specific directory update is available on the SAP Help Portal and is also included in each directory package. The directory update provides instructions for installing the directories and information about the directory contents. It describes the package names, files names, and the supported countries, directory format, geocoding mode, centroid level, and census information. It also lists the available POI types supported by the vendor.

3.5.4.10 Global Address Cleanse

The Global Address Cleanse transform identifies, parses, validates, and corrects global address data, such as primary number, primary name, primary type, directional, secondary identifier, secondary number, locality, region and postcode.

> Note
>
> The Global Address Cleanse transform does not support CASS certification or produce a USPS Form 3553. If you want to certify your U.S. address data, you must use the USA Regulatory Address Cleanse transform, which supports CASS.

If you perform both address cleansing and data cleansing, the Global Address Cleanse transform typically comes before the Data Cleanse transform in the data flow.

Sample transform configurations

The Global Address Cleanse transform has sample transform configurations that will help you to set up your data flow. The transforms include all of the required options except input fields.

Related Information

Transform configurations [page 1123]
Content objects [page 1133]
Address Cleanse reference [page 1379]

3.5.4.10.1 Global Address Cleanse transform options

3.5.4.10.1.1 Report And Analysis options

Choose to generate report data for the Global Address Cleanse transform.

Option	Description
Generate Report Data	Specifies whether to generate report data for this transform. *Yes*: Generates report data for this transform. *No*: Turns off report data generation. If you do not need to generate reports (during testing, for example), you should set this option to No to improve performance.

Related Information

Global Address Cleanse [page 1197]

3.5.4.10.1.2 Reference files

Reference files are directories required by the Global Address Cleanse transform to process your data.

You can use a substitution variable for the directory location. The substitution variable for reference files is $ $RefFilesAddressCleanse. Use this substitution variable if you accepted the default installation directory when you installed Data Services. If you installed to a location that wasn't the default location, you can change the substitution variable dynamically.

For information about downloading international directories, see the latest directories update.

Also, see the “Directory data” topic in the *Installation Guide*.

3.5.4.10.1.3 Country ID options (Global Address Cleanse)

Specifies whether or not to use Country ID processing. This option group is required.

Option	Description
Country ID Mode	Specifies whether to always use the specified Country Name or to run Country ID processing. *Constant*: Assumes all of your input data is for the specified Country Name and does not run Country ID processing. Choose this option only if all of your data is from one country, such as Australia. This option may save processing time. *Assigned*: Runs Country ID processing. Choose this option if the input data contains addresses from more than one country.
Country Name	Specifies the country of destination. *None*: Select when the *Country ID Mode* is set to Assigned and you don't want a default country to be set when the country cannot be identified. Special considerations: ● If the *Country ID Mode* is set to Constant, choose the country of destination from the Country Name list. The transform assumes that all of your data is for this country. i Note You cannot choose None if the *Country ID Mode* option is set to Constant. ● If the *Country ID Mode* is set to Assigned, choose a country name to be used when the Country ID could not identify a country. ● If Country Name is set to None, then the address will be sent to the default engine, Global Address.
Script Code	Specifies the ISO four-character script code for your data. *CJKK*: Chinese, Japanese, and Korean *GREK*: Greek *LATN*: Latin *None*: If this option is selected, transform will attempt to identify the overall script of the input data as CJKK, GREK, LATN, or other.

3.5.4.10.1.4 Standardization options

These options are found under ▶ *Standardization Options* › *Country* › *Options* ◀.

The Options group includes all the options that you need to standardize your address data. These settings apply to the country that you specify for ▶ *Standardization Options* › *Country* › *Country Name* ◀. This option group is required.

> i Note
>
> You can set these options for all countries or by individual country. To add another Country options group, Right click ▶ *Standardization Options* › *Country* ◀ and select *Duplicate Option*.

Option	Description
Address Line Alias	Specifies how to standardize the address line. (Engines supported: Canada, Global Address, and USA) *Convert*: Converts address lines based on Official address line components instead of Delivery address line components. *Preserve*: Retains non-preferred data in address lines unless the data is incorrect.
Assign Locality	Specifies how to standardize the locality name. *Convert*: Converts the locality name to the locality name preferred by the country's postal authority. *Preserve*: Preserves the input locality name unless it is incorrect. *Valid*: Retains the input locality name unless it is not valid for mailing. If it is not valid for mailing, replaces it with the preferred locality name.
Capitalization	Specifies the casing of your address data. *Mixed*: Converts data to initial capitals. For example, "MAIN STREET SOUTH" becomes "Main Street South." *Upper*: Converts data to full capitals. For example, "Main Street South" becomes "MAIN STREET SOUTH." i Note If you want consistent casing for your data, make sure that this option and the Capitalization setting in the Data Cleanse transform are the same.
Character Width Style	Specifies whether to standardize half-width and full-width characters. This option only applies to Chinese and Japanese data. *Normal Width*: Converts full-width Latin characters to half-width and converts half-width Chinese and Japanese characters to full width. *Half Width*: Convert all characters to half width. *Full Width*: Convert all characters to full width.
Convert_Latin_Output_To_US_ASCII	For Latin script records, converts any extended ASCII characters in the Best component to US ASCII characters, if a character conversion is available. For example, with the input street name “Østerbrogade”, you can preserve the local character or convert it to the international data format “Osterbrogade” in the cleansed output. Any extended ASCII characters for which there is no conversion (such as the degree symbol or inverted exclamation and question marks), are left as is. By default, the option is set to No. *Yes*: Converts extended ASCII characters. *No*: Does not convert extended ASCII characters.

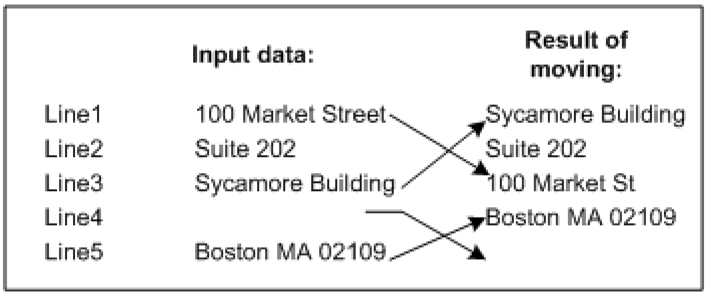
Option	Description
Correct Assigned Data	Specifies whether to use the parsed or corrected data for the assigned output fields of type Best. *Yes*: Populates the Best components with corrected data. *No*: Populates the Best components with parsed data. ℹ Note If you choose No for this option, the Capitalization option is the only available Standardization option for your assigned data.
Correct Unassigned Data	Specifies whether the Global Address Cleanse transform standardizes your unassigned data. *Yes*: Populates the Best components with corrected data. *No*: Populates the Best components with parsed data. ℹ Note If you choose No for this option, the Capitalization option is the only available Standardization option for your unassigned data.
Country Style	Specifies how to standardize the country data. *ISO_2CHAR*: Standardizes country data to the two-character ISO code, such as AU, CA, or US. *ISO_3CHAR*: Standardizes country data to the three-character ISO code, such as AUS, CAN, or USA *ISO_3DIGIT*: Standardizes country data to the three-digit ISO code, such as 038, 124, or 840. *Name*: Standardizes country data to the full country name, such as Australia, Canada, or United States. *Preserve*: Attempts to retain the country data in the input record, otherwise uses the corrected country value.
Directional Punctuation	Specifies whether to use punctuation in the abbreviated directional data. *Yes*: Outputs directionals with punctuation (for example, N. or S.W.) *No*: Outputs directionals without punctuation (for example, N, SW). *Preserve*: If there is punctuation in the input parsed component, returns the corrected punctuation if the input is not found in the dictionary (for example, N/W on input would be updated to N.W.).
Directional Style	Specifies whether to abbreviate directional data. *Long*: Uses fully-spelled directionals such as "North," "South," "East," "West." *Preserve*: Preserves the style used in the input record.

Option	Description
	Short: Uses abbreviated directionals such as "N," "S," "E," "W."
European Postcode Prefix	Adds the one- to three-character European Postcode prefix, followed by a dash, for mail generated and distributed inside Europe. *Yes*: Adds the European Postcode prefix. *No*: Does not add the European Postcode prefix. *Preserve*: Retains the European Postcode prefix, if one is found on input. In the following address, for example, the D- is the European Postcode extension. Hallesches Ufer 32-38 D-10963 Berlin Germany ℹ Note The European Postcode prefix is for mail distributed from one European country to another European country.
Extra Lines	Specifies what to do with extra lines of non-address data. *Preserve*: Attempts to retain the extra line of non-address data in the general location in which it was input. *Remove*: Does not include any extra line of non-address data in the standardized lines or multiline fields. *Preferred*: All populated Extra fields are placed above or below the multiline fields and standardized input lines based on the country data being processed. For example, Extra fields for Japan will be located below the standardized lines.

Format Assigned Data	Specifies whether to format your assigned data based on the country's preferred address format. For example, the format for Germany is:
	{Primary_Name1} {Primary_Number}
	{Postcode1} {Locality}
	{Country}
	Yes: Formats the assigned data.
	No: Does not format the assigned data and leaves it in the location in which it was input. If data is added to the record, this data will be placed based on the format string.

Format Unassigned Data	Specifies whether to format your unassigned data based on the country's preferred address format. For example, the format for Germany is:
	{Primary_Name1} {Primary_Number}
	{Postcode1} {Locality}

Option	Description
	{Country} *Yes*: Formats the unassigned data. *No*: Does not format the unassigned data and leaves it in the line in which it was input.
Include Country	Specifies whether to include country names in standardized lines or multiline fields. *Yes*: Includes country name. *No*: Does not include country name. *Preserve*: Retains the country name if found on input.
Include Locality Addition	Specifies whether the Locality1_Full output field contains both the Locality1_Name and the Locality1_Addition information. *Yes*: Includes both the locality and the locality addition information. *No*: Does not include the locality addition. *Preserve*: Includes locality addition if found on input. This is the default setting. Note The Locality Name Style option in the Global Address Cleanse transform overrides this option. If the Locality Name Style option is set to Short, the Locality1_Full field will not contain the locality addition information.
Include Unused Address Line Data	Specifies whether to output the unused address line data (for standardized lines and multiline fields). This option affects unused address data classified as remainder, but not unused address data classified as extra. *Yes*: Outputs the unused address line data in the remainder fields Address_Line_Remainder1 through Address_Line_Remainder4 (for example, 100 Main St Red House). *No*: Does not output the unused address line data (for example, 100 Main St).
Include Unused Lastline Data	Specifies whether to output the unused last line data (for standardized lines and multiline fields): *Yes*: Outputs the unused last line data. *No*: Does not output the unused last line data.
Locality Name Style	Specifies the format for locality data in the Locality1_Name output field for addresses. This option applies to German addresses. *Preserve*: Preserves the locality data format as it was input. This is the default setting. *Short*: Outputs locality data in the abbreviated version, if available in the reference data.

Option	Description
	Note To use the short locality name style, the *Address Line Alias* option must be set to *Convert*. **Note** This option overrides the Include Locality Addition option.
Move Multiline Data	Determines the position of blank lines in output addresses. *Bottom*: If there are any blank lines, the transform moves them to the top and shifts the data to the bottom of the address block. *No*: Does not rearrange any lines, blank or otherwise. *Top*: If there are any blank lines, the transform moves them to the bottom of the address block and shifts the data to the top of the block. Example of Top:  Input data: Result of moving: Line1 Line2 Line3 Line4 Line5 100 Market Street Suite 202 Sycamore Building Boston MA 02109 Sycamore Building Suite 202 100 Market St Boston MA 02109
Output Country Language	Specify which language and script to use on output for the country name (not the entire record). *Preserve*: Preserves country name as it was on input. *Catalan - Latin* *Chinese - Hani* *Danish - Latin* *Dutch - Latin* *English - Latin* *Finnish - Latin* *French - Latin* *Greek - Greek* *German - Latin*

Option	Description
	Hungarian - Latin *Italian - Latin* *Japanese - Hani* *Japanese - Kana* *Korean - Hang* *Norwegian - Latin* *Polish - Latin* *Portuguese - Latin* *Spanish - Latin* *Swedish - Latin*
Postal Phrase Punctuation	If you choose *Short* for the Postal Phrase Style option, this option specifies whether to use punctuation in the postal abbreviation. *Yes*: Includes punctuation in postal abbreviations (for example P.O. Box). *No*: Does not insert any punctuation for postal abbreviations (for example, PO Box). *Preserve*: If there is punctuation in the input parsed component, returns the corrected punctuation if the input is not found in the dictionary.
Postal Phrase Style	Specifies whether to abbreviate postal phrases. *Long*: Outputs the fully-spelled postal phrase (for example Post Office Box). *Preserve*: Retains the style of the postal phrase if found in the input record. *Short*: Outputs the abbreviated postal phrase (for example, PO Box). The punctuation for this option is determined by the Postal Phrase Punctuation option.
Primary Type Punctuation	If you choose *Short* for the Primary Type Style option, this option specifies whether to use punctuation in primary type abbreviations. *Yes*: Includes a period at the end of primary type abbreviations (for example, St.). *No*: Does not insert any punctuation at the end of primary type abbreviations (for example, St). *Preserve*: If there is punctuation in the input parsed component, returns the corrected punctuation if the input is not found in the dictionary.
Primary Type Style	Specifies the style for primary type address elements. *Long*: Uses fully spelled primary types such as Street, Avenue, Road, or Strasse. *Preserve*: Retains the style used in the input record *Short*: Uses abbreviated primary type such as St, Ave, Rd, or Str. The punctuation for this option is determined by the Primary Type Punctuation option.

Option	Description
Region Style	Specifies whether to abbreviate the region name (for example, state or province). *Long*: Uses the fully spelled region name (for example, California or Ontario). *Preserve*: Retains the style used in the input record. *Short*: Abbreviates the region name (for example, CA or ON).
Remove Address Apostrophes	Specifies whether to include apostrophes in certain street names that include a DE L' or D'. *Yes*: Retains apostrophes in street names if it was present on input, for example, Rue D'Abbeville. *No*: Removes apostrophes in street names, for example, Rue D Abbeville.
Secondary Description Punctuation	If you choose *Short* for the Secondary Description Style, this option specifies whether to use punctuation in the abbreviation. *Yes*: Uses punctuation in the abbreviation (for example, Apt.). *No*: Does not use punctuation (for example, Apt). *Preserve*: If there is punctuation in the input parsed component, returns the corrected punctuation if the input is not found in the dictionary.
Secondary Description Style	Specifies whether to abbreviate the secondary description (for example, a unit or an apartment). *Long*: Uses the fully spelled secondary description (for example, Apartment). *Preserve*: Retains the style used in the input record. *Short*: Abbreviates the secondary description (for example, Apt). The punctuation for this option is determined by the Secondary Description Punctuation option.
Secondary Number Style	Specifies the format of the secondary number (for example, a suite or apartment number). This option applies to Canada and New Zealand addresses. *Attached*: Converts all secondary ranges to the attached format, so that the secondary number is prepended to the primary number and separated with a delimiter. For example, for Canada addresses, it places a dash between the secondary and primary range: 5-100 Main St. *Preserve*: Preserves the style of the address as it was input. *Unattached*: Converts all secondary ranges to the unattached format. For example, for Canada, it places the unit designator at the end of the primary address: 100 Main St Suite 5.
Street Name Style	Specifies the format for street data for addresses. This option applies to addresses in Germany and the Netherlands. *Preserve*: Preserves the street data format as it was input. For example: Annelien Kappeyne Van de Coppellostr 2

Option	Description
	Herten Limburg 6049 HD *Short*: Outputs street data in the format preferred by the postal authority. For the Netherlands, this returns a street address with a maximum of 24 characters in mixed case. For example: A K vd Coppellostr 2 Herten Limburg 6049 HD Note To use the short street name style, the *Address Line Alias* option must be set to *Convert*. Note The *Capitalization* option in the Global Address Cleanse transform overrides the *Street Name Style* option.
Use Local Primary Type Style	Specifies whether to use the type style for primary address components that is present in the address data. Setting this option to *Yes* ignores the *Primary Type Style* option. This option applies to Austria, Germany, and Switzerland. *Yes*: Uses the Primary Type Style that is present in the address data. *No*: Uses the Primary Type Style specified in the Primary Type Style option.
Use Postal Country Name	Specifies which country data is output for countries that receive their postal service from another country. For example, if you are using the USA engine and have addresses from the U.S. territories, the Country field is populated with the postal country (United States) rather than the territory name (such as American Samoa, Puerto Rico, and so on). The style of the Country field is still based on the Country Style option. If the country does not have a postal country, this option does not change the output. *Yes*: Uses the postal country name. *No*: Uses the territory country name.

Related Information

Canada engine [page 1209]

Global Address engine options [page 1214]
USA engine [page 1218]

3.5.4.10.1.5 Engines

Assigns the engines that you want to use with the Global Address Cleanse transform.

The Global Address Cleanse transform must always have one or more of the Global Address Cleanse engines enabled in order to process your data.

This option group is required.

Option	Description
Canada	Specify whether the Canada engine is used for the Global Address Cleanse transform. *Yes*: Activates the Canada engine. *No*: De-activates the Canada engine.
Dynamic Engine Init	Specify whether engines that are used for the Global Address Cleanse transform are limited to the engines that can be initialized. *Yes*: Uses only the engines that can successfully be initialized. If an engine fails to initialize, a warning is issued and the job continues. *No*: Uses all enabled engines. If an engine fails to initialize, the job fails. This is the default value.
Global Address	Specify whether the Global Address engine is used for the Global Address Cleanse transform. *Yes*: Activates the Global Address engine. *No*: De-activates the Global Address engine.
USA	Specify whether the USA engine is used for the Global Address Cleanse transform. *Yes*: Activates the USA engine. *No*: De-activates the USA engine.

Related Information

Canada engine [page 1209]
Global Address engine options [page 1214]
USA engine [page 1218]

3.5.4.10.1.5.1 Canada engine

Use the Canada engine to process your Canada address data with the Global Address Cleanse transform. The engine includes specific options that you can set for processing Canada address data and suggestion lists.

Related Information

Reference files [page 1198]
Canada engine options [page 1209]
Canada engine Suggestion List options [page 1212]
Canada engine Report options [page 1212]

3.5.4.10.1.5.2 Canada engine options

The Options group contains all of the specific settings that you must define when processing with Canada address data.

Option	Description
Disable Certification	*Yes*: Enables non-certified processing of Canada addresses and allows processing with non-POC (Point-of-Call) directories for non-mailing purposes. When you select *Yes*, you cannot print the SERP Report. Any list created with certification disabled cannot be used for mailing. *No*: Enables certified processing for Canada using POC (Point-of-Call) directories and enables printing of the SERP report. Note If this option is set to *Yes*, the following options should also be set to *Yes*: ● Enable LVR Rule ● Enable Rural Rule ● Postcode Only Search ● Postcode No Match Search
Dual Address	Specifies the action to take when the Canada engine encounters a dual address. *Position:* Selects an address based on the arrangement of the input data. The Canada engine tries to validate the address that is closest to the lower left corner of the address block. That might be the postal or the street address, depending on how the data was entered. (This value is required for SERP certification.) *Postal:* Tries to validate based on the postal address. If that fails, tries again based on the street address.

Option	Description
	Street: Tries to validate based on the street address. If that fails, tries again based on the postal address (rural route or PO box).
Enable LVR Rule	Canada Post requires that any address with a valid Large Volume Receiver (LVR) postal code be considered valid. The postal code cannot be changed to match other address components. Canada Post recommends that you don't correct LVR addresses; however, correction is permitted when a unique address can be determined without changing the postal code. *Yes:* Regards any LVR address as assigned, even when the address line is so flawed that a match to the postal directory is impossible. (This value is required for SERP certification.) *No:* Disables this rule. The transform reports an LVR address as unassigned when the address line is flawed.
Enable Rural Rule	Canada Post requires that any address with a valid rural postal code must be considered valid. (Rural postal codes have a zero in the second position.) This rule applies even if the address line is empty or contains bad data. Canada Post recommends that you don't correct rural addresses; however, the Canada engine will always attempt to correct the rest of the address. The valid rural postal code will always be left intact, according to CPC rules. This also applies if an address is entered without a postal code or with an incorrect postal code, and the locality (city) entered has just one postal code associated with it that is a rural postal code. *Yes:* Regards any rural address as valid, even if the address line is so flawed that a match to the postal directory is impossible. (This value is necessary for SERP certification). *No:* Reports a rural address as invalid if the address line is bad.
Output Address Language	*Convert:* Uses French for records in Quebec, and English for records in other regions (provinces). *English:* Converts records to English. *French:* Converts records to French. *Preserve:* Detects the input language and preserves that language upon output, no matter the region (province).
Parse Only	*Yes:* Parses records into discrete components, but does not perform a lookup in the postal directories. Parse Only is fast, but parsing results are unverified. *No:* Parses records into discrete components and performs a lookup in the postal directories. Setting this option to No may slow down processing, but parsing results are verified when the appropriate reference dat is available.
Postcode No Match Search	This option is important when the Canada engine has determined that the address line can be assigned, but doesn't match the incoming postal code. SERP rules specify that if this occurs, the transform must search the postal directories to ensure the following:

Option	Description
	- If the incoming address line is a PO Box address, the postal code must not be a valid postal code for an LVR (Large Volume Receiver), firm, or a civic (street) address, such as 100 Main St. - If the incoming address line is a civic (street) address, the postal code must not be a valid postal code for an LVR PO Box address. If either one of these conditions exist, Data Services cannot assign the address, according to SERP rules. Because doing a postal-code-only search is very time consuming, disabling this search should speed up your processing time. *Yes:* Turns on this option. (This value is necessary for SERP certification.) *No:* Turns off this option.
Postcode Only Search	This option affects assignment when the input address line is badly incomplete (for example, when the address includes a range but no street name). In this case, SERP rules specify that the transform must search based on postal code only, and attempt to find a street record containing that range. If the Canada engine can find only one street record that contains the range, then (the SERP rules state) the address line is assigned from the postal code. *Yes:* Turns on the option. (This value is necessary for SERP certification.) *No:* Turns off the option. In some cases, the result is a better address line. In other cases, the Canada engine more reliably detects that it cannot assign an address line.
Postcode Priority Over Street	Specifies whether postcode and street are equally weighted or whether street has priority over postcode. This option is important when the Canada engine is trying to break a tie between two possible assignments: - A near match on address line. - An exact match on postal code. *Yes:* When breaking a tie between a near match on street address line and an exact match on postal code, validates by placing more weight on the address line. Where possible, the transform changes the postal code to agree with the address line, because data-entry errors are common in postal codes. *No:* When breaking a tie between a near match for the address line and an exact match for the postal code, places an equal weight on the address line and the postal code. (This value is necessary for SERP certification.)
Unit Description	Specifies the unit description in English: *Apartment*: Uses Apartment as the default unit designator. *Default*: Uses the default unit designator. *Unit*: Uses Unit as the default unit designator.

Related Information

Standardization options [page 1199]

3.5.4.10.1.5.3 Canada engine Report options

Set these options to add the necessary Statement of Address Accuracy report information.

This is an optional group, however this option group must be completed so that Data Services produces a SERP Report (Software Evaluation and Recognition Program).

Option	Description
Customer Company Name	Specifies the company name of the organization for whom you are preparing the mailing (up to 40 characters).
Mailer Address1 Mailer Address2 Mailer Address3 Mailer Address4	Specifies the name and address of the person or organization for whom you are preparing the mailing (up to 40 characters per line).
Customer CPC Number	Specifies the customer's CPC number that is located on the Canada Post Contract (up to 15 characters).

3.5.4.10.1.5.4 Canada engine Suggestion List options

Set these options when you want to generate suggestion lists for your Canada address data.

Option	Description
Address Lines Match Minimum	Specifies the similarity score required for address-line suggestions. This score then determines which suggestions will be returned in the list. A higher number indicates that the suggestion must be more similar to the input in order to be returned as a possible suggestion. Type a value from 0 to 80.
Address Range	Specifies a span around the input primary address range for which to return suggestions. By using this option, you can limit the suggestions returned to be within a few blocks of your input. For example, assume you entered 500 for this value. Then, you submit the following street address: 1000 Pine St. Suggestions would be returned in a range from 750 to 1250 Pine Street.

Option	Description
	If you don't want to limit the ranges returned in suggestions, type 0.
Combine Overlapping Ranges	Specifies whether individual suggestions with overlapping ranges are combined. *Yes:* Ignores gaps and overlaps in ranges. Set this option to Yes if you want to limit the number of total suggestions presented to your user. However, you might not see gaps of invalid ranges that would be apparent if this option was set to No. For example, the following suggestions would be presented if this option is set to No: 1000-1099 Maple Ave 1100-1199 Maple Ave But this suggestion would only show if set to Yes: 1000-1199 Maple Ave *No:* Does not combine overlapping ranges.
Enable Suggestion Lists	Specifies whether suggestion lists are generated. *No:* Does not generate suggestion lists. *Yes:* Generates suggestion lists when assignment candidates are present.
Lastlines Match Minimum	Specifies the similarity score required for lastline suggestions. This score then determines which suggestions will be returned in the list. A higher number indicates that the suggestion must be more similar to the input in order to be returned as a possible suggestion. Type a value from 0 to 80.
Max Number Address Lines	Specifies the maximum number of address line suggestions that can be generated. You might set this option in order to limit the size of the SOAP documents being sent by the web service, or to limit the maximum number of suggestions that your users would have to choose from. However, by setting a maximum, you may occasionally eliminate a suggestion from the list that could be the correct one. The minimum number you can enter is 2. The maximum number you can enter is 50.
Max Number Lastlines	Specifies the maximum number of lastline suggestions that can be generated. You might set this option in order to limit the size of the SOAP documents being sent by the web service, or to limit the maximum number of suggestions that your users would have to choose from. However, by setting a maximum, you may occasionally eliminate a suggestion from the list that could be the correct one. The minimum number you can enter is 2. The maximum number you can enter is 15.

3.5.4.10.1.5.5 Global Address engine options

The Global Address engine includes specific options that you can set for processing global address data and suggestion lists.

Option	Description
Country Name	Choose a specific country for the Assignment option settings or choose Global (Apply to All Countries) to make global settings.
Disable Certification	Specifies whether to perform non-certified or certified processing of addresses for Australia, France, or New Zealand. **Note** If the reference data is missing or older than the certification requirements for any country (Australia, France, or New Zealand), a warning message will be issued. If the message is issued for data that is not required for the country you are processing, you can ignore the message. To avoid receiving this message, replicate the country options group for each country (Australia, France, and New Zealand) and only set the *Disable Certification* option to *No* for the country data you are processing. When you set the options for each country, all other options in the country-specific group will be used in place of the options selected in the global group. *Australia*: ● *Yes*: Enables non-certified features and extends the directory expiration for non-mailing purposes. You may extend the directory expiration period up to 14 months from the date the directories were created. Processing with expired directory data is allowed when you are not planning to use the records for AMAS mailing purposes. This is ideal for data warehousing industries, for example. However, when you select Yes, you cannot print the AMAS report. Any lists created with expired directories cannot be used for postage discounts. Data directories expire after 15 months when certification is disabled. ● *No*: Uses the most current directory information, disables non-certified features, and enables printing of the AMAS report. *France*: ● *Yes*: Enables non-certified processing of France addresses for non-mailing purposes. ● *No*: Enables certified processing of France addresses for mailing purposes. All punctuation will be removed except for firm data. Accented characters will have their accents removed (only A-Z and 0-9 are allowed). The returned address will be in 6 lines. **Note** To return the address in all upper case, set the Capitalization option under ▸ *Standardization Options* ▸ *Country* ▸ *Options* ◢ to *Upper*. *New Zealand*:

Option	Description
	• *Yes*: Enables non-certified processing of New Zealand addresses and allows processing with expired directories for non-mailing purposes. When you select Yes, you cannot print the SOA Report. Any list created with certification disabled cannot be used for mailing. • *No*: Enables certified processing for New Zealand and enables printing of the SOA report.
Dual Address	Specifies the action to take when the Global Address engine encounters a dual address. *Position*: Selects an address based on the arrangement of the input data. The transform attempts to assign the first address found in the input data. If it cannot assign the first address, it will attempt to assign the next address found. *Postal*: Attempts to validate based on the postal address. If that fails, attempts again based on the street address. *Street*: Attempts to validate based on the street address. If that fails, attempts again based on the postal address.
Output Address Script	Specifies how a native script is output. Supports Chinese (if the input character script is CJKK) and Russian (if the input character script is CYRL). *Latin*: Transliterates the native script and outputs the data as Latin. *Preserve*: Outputs the data in the native script.
Retain Postcode if Valid Format	Set this option for processing valid or invalid input postcodes. *Yes*: Retains input postcode unless it is an invalid format for the country and there is a single answer or intelligent matching is possible. *No*: Updates output postcode if there is a single answer or intelligent matching is possible. Otherwise retain input postcode. Note If the postcode is blank, the transform will add one if there is a single answer.

3.5.4.10.1.5.6 Global Address engine Suggestion List options

Set these options when you want to generate suggestion lists for your global address data.

Option	Description
Address Range	Specifies a span around the input primary address range for which to return suggestions. By using this option, you can limit the suggestions returned to be within a few blocks of your input. For example, assume you entered 500 for this value. Then, you submit the following street address: 1000 Pine St.

Option	Description
	Suggestions would be returned in a range from 750 to 1250 Pine Street. If you don't want to limit the ranges returned in suggestions, type 0. The maximum value is 5000.
Combine Overlapping Ranges	Specifies whether individual suggestions with overlapping ranges are combined. *Yes:* Ignores gaps and overlaps in ranges. You might set this option to Yes if you want to limit the number of total suggestions presented to your user. However, you might not see gaps of invalid ranges that would be apparent if this option was set to No. For example, a suggestion list might show the following suggestions if this option is set to No: 1000-1099 Maple Ave 1100-1199 Maple Ave But would only show this suggestion if set to Yes: 1000-1199 Maple Ave *No:* Does not combine overlapping ranges. ℹ Note It is recommended that if the option is set to *No*, you should consider increasing the size of the Suggestion_List output field so that it can contain all of the suggestions.
Enable Suggestion Lists	Specifies whether suggestion lists are generated. *No*: Does not generate suggestion lists. *Yes*: Generates suggestion lists when assignment candidates are present.
Max Number Address Lines	Specifies the maximum number of address-line suggestions that can be generated. You might set this option in order to limit the size of the SOAP documents being sent by the web service, or to limit the maximum number of suggestions that your users would have to choose from. The minimum number you can enter is 2. The maximum number you can enter is 10,000. The default setting is 100.
Max Number Lastlines	Specifies the maximum number of lastline suggestions that can be generated. You might set this option in order to limit the size of the SOAP documents being sent by the web service, or to limit the maximum number of suggestions that your users would have to choose from.

Option	Description
	The minimum number you can enter is 2. The maximum number you can enter is 10,000. The default setting is 60.

3.5.4.10.1.5.7 Global Address engine Report options

With the Report Options group, you can add the required information for the following reports:

- *Report options for New Zealand* [page 1217]
- *Report options for Australia* [page 1217]

3.5.4.10.1.5.8 Report options for New Zealand

With the Report Options group, you can add the required Statement of Accuracy (SOA) report information.

Option	Description
Customer Company Name	Specifies the name of the customer company name for whom you are preparing this list (up to 40 characters).
Customer Number	New Zealand Post customer number. If you want to submit your file for mailing and qualify for postage discounts, you must include your customer number on the report.
File Name	Name of the input data associated with the report.
Mailer Address 1 Mailer Address 2 Mailer Address 3 Mailer Address 4 Mailer Address 5 Mailer Address 6	Specifies the name and address of the person or organization for whom you are preparing the mailing (up to 29 characters per line).
SOA Issuer Name	Specifies the name of the company that prepared this list (up to 40 characters).

3.5.4.10.1.5.9 Report options for Australia

With this option group, you can add the required Australia AMAS - Address Matching Processing Summary information.

This is an optional group, however this option group must be completed so that Data Services produces an AMAS Report (Address Matching Processing Summary Report).

Option	Description
Customer Company Name	Specifies the name of the customer company name for whom you are preparing this list (up to 40 characters).
List Name	Specifies the name of your database or mailing list (up to 40 characters). This might be the file name, your title, or formal name for the list.
File Name	Specifies the actual input file name, such as australia.dbf (up to 40 characters).
Mailer Address1 Mailer Address2 Mailer Address3 Mailer Address4	Specifies the name and address of the person or organization for whom you are preparing the mailing (up to 29 characters per line).

3.5.4.10.1.5.10 USA engine

Use the USA engine with the Global Address Cleanse transform to address cleanse your data for the United States of America and its territories. The engine includes specific options that you can set for processing USA data.

Related Information

USA engine Suggestion List options [page 1219]
Standardization options [page 1336]

3.5.4.10.1.5.11 USA engine options

The Options group contains all of the specific settings that you must define when processing with USA address data.

Option	Description
Dual Address	Specifies the action to take when the transform encounters a dual address. *Position*: Selects an address based on the arrangement of the input data.

Option	Description
	The transform attempts to validate the address that is closest to the lower left corner of the address block. That might be the postal or the street address; it depends on how the data was entered. *Postal*: Attempts to validate based on the postal address. If that fails, attempts again based on the street address. *Street*: Attempts to validate based on the street address. If that fails, attempts again based on the postal address (rural route or PO Box).
Parse Only	*Yes*: Parses records into discrete components, but does not perform a lookup in the postal directories. Parse Only is fast, but parsing results are unverified. *No*: Parses records into discrete components and performs a lookup in the postal directories. Setting this option to No may slow down processing, but parsing results are verified.
Unit Description	Specifies how to standardize the unit description. *Convert*: Uses the unit description found in the postal directory (such as an apartment, suite, room, or floor). *Preserve*: Preserves the unit description from the input record, correcting any spelling errors.

Related Information

Standardization options [page 1199]

3.5.4.10.1.5.12 USA engine Suggestion List options

Set these options to generate suggestion lists for the USA and its territories.

Option	Description
Address Lines Match Minimum	Specifies the similarity score required for address-line suggestions. This value then determines which suggestions will be returned in the list. A higher number indicates that the suggestion must be more similar to the input in order to be returned as a possible suggestion. Type a value from 0 to 80.
Address Range	Specifies a span around the input primary address range for which to return suggestions. By using this option, you can limit the suggestions returned to be within a few blocks of your input. For example, assume you entered 500 for this value. Then, you submit the following street address:

Option	Description
	1000 Pine St. Suggestions would only be returned in a range from 750 to 1250 Pine Street. If you don't want to limit the ranges returned in suggestions, type 0.
Combine Overlapping Ranges	Specifies whether individual suggestions with overlapping ranges are combined. *Yes:* Ignores gaps and overlaps in ranges. You might set this option to Yes if you want to limit the number of total suggestions presented to your user. However, you might not see gaps of invalid ranges that would be apparent if this option was set to No. For example, a suggestion list might show the following suggestions if this option is set to No: 1000-1099 Maple Ave 1100-1199 Maple Ave But would only show this suggestion if set to Yes: 1000-1199 Maple Ave *No:* Does not combine overlapping ranges.
Enable Suggestion Lists	Specifies whether suggestion lists are generated. *Yes:* Generates suggestion lists when assignment candidates are present. *No:* Does not generate suggestion lists.
Lastlines Match Minimum	Specifies the similarity score required for lastline suggestions. This score then determines which suggestions will be returned in the list. A higher number indicates that the suggestion must be more similar to the input in order to be returned as a possible suggestion. Type a value from 0 to 80.
Max Number Address Lines	Specifies the maximum number of address line suggestions that can be generated. Limits the size of the SOAP documents being sent by the web service, or limits the maximum number of suggestions that your users would have to choose from. However, by setting a maximum, you may occasionally eliminate a suggestion from the list that could be the correct one. The minimum number you can enter is 2. The maximum number you can enter is 100.
Max Number Lastlines	Specifies the maximum number of lastline suggestions that can be generated. Limits the size of the SOAP documents being sent by the web service, or limits the maximum number of suggestions that your users would have to choose from. However, by setting a maximum, you may occasionally eliminate a suggestion from the list that could be the correct one. The minimum number you can enter is 2. The maximum number you can enter is 100.

3.5.4.10.1.6 Suggestion List (Global Address Cleanse transform)

The Suggestion List option group contains three options for constructing the suggestion list string. The string contains components based on the suggestion type that you generate. Each suggestion list option group contains fields associated with the suggestion type. The first table lists the Suggestion List group, and the second table lists the Suggestion Type fields.

Table 158: Suggestion List group

Option	Description
Suggestion List Components	Choose the address component fields that you want to include in the address suggestions.
Delimiter	Specifies a character to use to separate each suggestion list selection. This value is only considered if the Output Style option is set to Delimited. This value can be any character or string. Common delimiters include a pipe symbol (|), or a string of multiple asterisks (***). This value should differ from the Field Delimiter value.
Field Delimiter	Specifies a character to use to separate each field in a single suggestion list. Each selection can be made up of one or more fields. If you choose to retrieve multiple fields per selection, those fields are separated with the character you specify here. This value can be any character string. The default value is a pipe symbol (|). This value should differ from the Delimiter value.
Output Style	Specifies the format for how the suggestion list data should be output. *Delimited:* Outputs the suggestion list data in a delimited text format, with the delimiters specified in the Delimiter and Field Delimiter options. *XML:* Outputs the suggestion list data as hierarchical XML. This option is likely the preferred one for users who integrate suggestion lists via the web service. You can then use the XML tools you own to parse the suggestion list data.

Table 159: Suggestion List components

Field	Description
Building_Name	The building name for the address, which in some countries is used in place of the primary number. For example, in the U.K. an address may be

Field	Description
	“White House, High Street,” where “White House” is the building name instead of a primary number in an address such as “100 High Street.” In some cases, an address will include the bulding name and primary number.
Delivery_Installation_Name	The delivery installation city name, which is usually the same as the city name and (if it is the same) omitted from the address line.
Delivery_Installation_Qualifier	Delivery Installation qualifier (for example, “Main” in “RR 2 Vancouver Stn Main”).
Delivery_Installation_Type	The delivery installation type. *English*: *PO*: Post Office. *RPO*: Retail Post Outlet. *STN*: Station. *LCD*: Letter Carrier Depot. *CMC*: Community Mail Center. *CDO*: Commercial Dealership Outlet. *French*: *BDP*: Bureau de Poste. *CSP*: Comptoir Service Postal. *SUCC*: Succursale. *PDF*: Poste de Facteurs. *CPC*: Centre Postal Communautaire. *CC*: Concession Commerciale.
Firm	Returns the firm, company, or organization name.
Floor_Description	The level description, such as “Floor.”
Floor_Number_High Floor_Number_Low	If the floor number is a range such as 20-22, LOW contains “20” and HIGH contains “22.” If the floor number is not a range, both fields contain the floor number (for example, “20” and “20”).
Locality1 Locality2 Locality3 Locality4	Returns the city, town, or suburb. Additional locality information goes in Locality4.
Locality1_Official Locality2_Official	The locality name preferred by the postal authority.

Field	Description
Locality3_Official Locality4_Official	
Postcode	Returns the postal code.
Postcode1	*Australia*: Four-digit postcode. *Canada*: First three characters (FSA) of the postal code. *Global*: Postal code. *USA*: Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4).
Postcode2	The secondary postal code. *Canada*: The last three characters (LDU) of the postal code. *USA*: The four-digit ZIP Code, which on a mail piece, this code follows the primary postal code with a hyphen placed between (for example, 54601-1234).
Primary_Name1 Primary_Name2 Primary_Name3 Primary_Name4	Returns the street description. For example, Primary Name1 may return "Marina" and Primary Name2 may return "The Slipway." Suggestion List group: Available in the Primary Name Components option.
Primary_Name_Full1 Primary_Name_Full2	The primary name, primary type, primary prefix, and primary postfix.
Primary_Name_Full3 Primary_Name_Full4	The primary name and primary type.
Primary_Number_Description	A description preceding the primary number. For example, LOT (Australia).
Primary_Number_Extra	Data found near the parsed primary number, which in most cases cannot be identified or does not belong in a standardized address.
Primary_Number_Full	The primary number, primary number description, and primary number extra.
Primary_Number_High Primary_Number_Low	If the house number is a range such as 100-102, LOW contains “100” and HIGH contains “102.” If the house number is not a range, both fields contain the house number (for example, “100” and “100”).
Primary_Postfix1 Primary_Postfix2	Abbreviated or non abbreviated directional (for example, N, South, NW, SE) that follows a street name.

Field	Description
Primary_Prefix1 Primary_Prefix2	Abbreviated or non abbreviated directional (N, South, NW, SE) that precedes a street name.
Primary_Side_Indicator	Indicates if even, odd, or both values are valid. This applies to Street and PO box. *E*: The record is even-numbered. *O*: The record is odd-numbered. *B*: The record covers both the even- and odd-numbered sides of the street and PO Boxes.
Primary_Type1 Primary_Type2 Primary_Type3 Primary_Type4	The type of primary name (Some examples are rue, strasse, street, Ave, or Pl).
Region1–2	Returns the state, province, or region.
Secondary_Side_Indicator	Indicates if even, odd, or both values are valid. This applies to floors and units. *E*: The secondary record is even-numbered. *O*: The secondary record is odd-numbered. *B*: The secondary record covers both the even and odd-numbered values.
Selection	Returns a unique index number that identifies this suggestion from the others in the returned list. The suggestion “selection” number ranges from 1 to the number of suggestion selections in the suggestion list.
Stairwell_Description	Entrance or stairwell identifier for a building, such as, Stiege.
Stairwell_Name	The name or number of an entrance or stairwell for a building, such as, Stiege "1."
Unit_Description	Identifies a unit within a building, such as Room, Unit, Apt, Suite, and so on.
Unit_Number_High Unit_Number_Low	If the unit number is a range such as 1-20, Unit Number Low contains 1 and Unit Number High contains 20.

3.5.4.10.2 Global Address Cleanse fields

The Global Address Cleanse transform requires that you map fields on input and output.

3.5.4.10.2.1 Field category columns in Output tab

The Output tab lists output fields that hold the data that the transform cleanses or creates. You can choose to view the Best Practice, In Use, or All output fields by selecting the corresponding option at the top of the tab.

Best Practice: Lists all available output fields that have a field class of Best.

In Use: Lists only the output fields that you have chosen to output (listed in Schema Out).

All: Lists all output fields that are available for this transform.

> Note
>
> For details about mapping input and output fields, see the *Designer Guide*.

The output field attributes in the table below are listed in groups based on the field category column. Each field has categories that describe the type of content that is output. The field category displays "None" when it does not apply to the field.

Category	Description
Content Type	Identifies the type of data in the field. Setting the content type helps you map your fields when you set downstream transforms.
Field Addrclass	Specifies the address class for the generated field. *Delivery:* When used with the applicable Field Name, this value generates fields that reflect the address that is used in an attempt to assign an address. *Dual:* When used with the applicable Field Name, this value generates fields that reflect the address that is not used in an attempt to assign an address for input records that may contain both a street and postal address on input. *Official:* When used with the applicable Field Name, this value generates fields in the form of the data preferred by the Postal Authority. For example, in Winona, Minnesota USA, Broadway and 6th Street are alternate names for the same street. A letter addressed to Broadway is delivered, but the USPS prefers 6th Street.
Field Category	*Component:* Individual address components and postal codes that are related to the processed record. *Standardized:* Standardized input lines based on the settings in the Standardization Options group in the transform. *Suggestion:* Suggestion list output data based on the settings in the Suggestion List Options group.
Field Class (USA Regulatory Address Cleanse)	Specifies the field class that you want to assign to your output fields.

Category	Description
	Best: Outputs data based on various factors, such as whether an address was assigned, the Field AddrClass, and any settings that you defined in the Standardization Options group in the Options tab. Note When NCOALink is enabled and a valid move is available, Best fields contain the move-updated address data if it exists and if it matches in the U.S. National Directories. Or, the field contains the original address data if a move does not exist or if the move does not match in the U.S. National Directories. *Correct:* Outputs the complete and correct value found in the directories, and is standardized according to any settings that you defined in the Standardization Options group in the Options tab. *Parsed:* Outputs the parsed value. *Pre_LACSLink*: Retained address components that were replaced with LACSLink address information. *Move_Updated*: Outputs the address components that have been updated with move-updated address data. Note The transform looks for the move-updated address information in the U.S. National Directories. When the move-updated address is not found in the U.S. National Directories, the software populates the Move Updated fields with information found in the Move Update Directories only. The Move Updated fields that are populated as a result of standardizing against the U.S. National Directories is not updated.
Field Class (Global Address Cleanse)	Specifies the field class that you want to assign to your output fields. *Best:* Outputs data based on various factors, such as whether an address was assigned, the Field AddrClass, and any settings that you defined in the Standardization Options group in the Options tab. *Parsed:* Outputs the parsed value.
Field Name	Specifies a field name where the data is populated based on the options that you specify within this transform.
Type	Specifies the type and default length of data the output field contains; for example, varchar, date, and time.

Related Information

Content types [page 1131]
Designer Guide: How address cleanse works [page 586]

3.5.4.10.2.2 Input fields

The following are recognized Data Services input fields that you can use in the Global Address Cleanse transform. The table also shows that each input field is available based on the engine(s) that you enable:

- Canada (C)
- Global Address (G)
- USA (U)

See the fields listed in the transform's Input tab to view each field's properties.

Field	Description	Engine(s)
Address_Line	The delivery address line, for example, "123 Main Street, Unit 4." *China*: Address_Line may represent the following address components: ● Street and street number ● Building, floor, unit ● Residential community For example, 晨晖路 123 号中华大厦 12 楼 1201 室 宝山新村 100 号 201 室 *Japan*: Address_Line may represent the following address components: ● Block (chome, kumi, Hokkaido go), sub-block (banchi, gaiku, tochi kukaku), and house number (go) parts of the Japanese address. ● The building name, building floor, building room parts of the Japanese address. ● The P.O. Box portion of the address, if applicable.	All engines
Country	The identified country name of the address.	All engines
Data_Source_ID	Specifies the input source. This field is used in reports to identify the record.	All engines

Field	Description	Engine(s)
Firm	The name of a company or organization. In some countries, large firms have their own postal code. If you include a Firm field in your input, this transform may assign more specific postal codes. *China*: China does not support Firm assignment. There is no Firm data for China. If the Firm is available on input, place it in this field. *Japan*: All Firm data for addresses in Japan should be placed in this field.	All engines
Lastline	The locality, region (when it is included in the data), and postal code on one line.	All engines
Locality1	The city, town, or suburb. *China*: The Prefecture level localities. Prefectures (地区 diqu), Autonomous prefectures (自治州 zizhizhou), Prefecture-level cities (地级市 dijishi), Leagues (盟 meng), or Provincial countries (省直辖县 shengzhixiaxian). *Japan*: The city (shi), island (shima), ward (ku), county (gun) district (machi) or village (mura).	All engines
Locality2	Any additional city, town, or suburb information. *China*: County level localities, counties (县 xian), autonomous counties (自治县 zizhixian), county-level cities (县级市 xianjishi), districts (市辖区 shixiaqu), banners (旗 qi), autonomous banners (自治旗 zizhiqi), forestry areas (林区 linqu), or special districts (特区 tequ). *Japan*: Any additional ward, district, village or sub-district (aza, bu, chiwari, sen). *USA*: The Puerto Rican urbanization.	G, U
Locality3	Any additional city, town, or suburb information. *China*: Township level localities, townships (乡 xiang), ethnic townships (民族乡 minzuxiang), towns (镇 zhen), subdistricts (街道办事处 jiedaobanshichu), district public offices (区公所 qugongsuo), sumu (苏木 sumu), or ethnic sumu (民族苏木 minzusumu). *Japan*: Any additional district, village, sub-district (aza, bu, chiwari, sen, donchi, and tori), or super block (joh).	G
Multiline1-12	A line that may contain any data. The type of data in this line may vary from record to record.	All engines

Field	Description	Engine(s)
	Japan: Represents the lines that may contain any data with the following restrictions. The address in total has to be in the traditional order of a Japanese address. In addition, the block (chome, kumi, Hokkaido go), sub-block (banchi, gaiku, tochi kukaku), and house number (go) should be within one line on input.	
NetWeaver_Building	A specialized input field that contains the building information. If you map this input field, you must also map NetWeaver_Street. ⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	All engines
NetWeaver_Floor	A specialized input field that contains the floor number. If you map this input field, you must also map NetWeaver_Street. ⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	All engines
NetWeaver_House_Num1	A specialized input field that contains the house number. If you map this input field, you must also map NetWeaver_Street. ⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	All engines
NetWeaver_House_Num2	A specialized input field that contains additional house number information. If you map this input field, you must also map NetWeaver_Street.	All engines

Field	Description	Engine(s)
	⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	
NetWeaver_Location	A specialized input field that contains additional street information. If you map this input field, you must also map NetWeaver_Street. ⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	All engines
NetWeaver_PO_Box	A specialized input field that contains the PO Box number. ⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	All engines
NetWeaver_RoomNumber	A specialized input field that contains the room number. If you map this input field, you must also map NetWeaver_Street. ⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	All engines
NetWeaver_Str_Suppl1	A specialized input field that contains additional street information. If you map this input field, you must also map NetWeaver_Street.	All engines

Field	Description	Engine(s)
	⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	
NetWeaver_Str_Suppl2	A specialized input field that contains additional street information. If you map this input field, you must also map NetWeaver_Street. ⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	All engines
NetWeaver_Str_Suppl3	A specialized input field that contains additional street information. If you map this input field, you must also map NetWeaver_Street. ⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	All engines
NetWeaver_Street	A specialized input field that contains the primary street name. ⚠ Caution Use this field properly to avoid unexpected results in your data. For more information, see *Mapping NetWeaver input fields* [page 1232].	All engines
Postcode	The postal code. *USA*: The five-digit ZIP Code and ZIP+4.	All engines
Region1	The state, province, or region. *China*: Province-level regions, provinces (省 sheng), autonomous regions (自治区 zizhiqu), municipalities (直辖市 zhixia-	All engines

Field	Description	Engine(s)
	shi), or special administrative regions (特别行政区 tebie xingzhengqu). *Japan*: Represents the prefecture (to, do, fu, ken). A prefecture is similar to a state in the U.S.	
Region2	The state, province, or region.	All engines
Suggestion_Reply1-6	Used to input the index number that corresponds to a specific last line suggestion, an address line suggestion, or secondary list suggestion. These fields can also be used to input a street primary range or a street secondary range. *Suggestion_Reply1*: If you do not want to use a suggestion list, make the value of this field 0 and the suggestion list will be ignored.	All engines

3.5.4.10.2.3 Mapping NetWeaver input fields

The NetWeaver input fields are designed to be used in conjunction with other NetWeaver input fields in SAP software. Use the fields properly to avoid unexpected results in your data.

You cannot map multiline or Address_Line input fields when you use the NetWeaver input fields. Although the NetWeaver input fields appear discrete, they behave and are processed as multiline fields. They are mapped internally to Multiline1-12 before normal Global Address Cleanse processing is performed. If a NetWeaver input field is not mapped, the multiline that would have been mapped to it is mapped to the next available NetWeaver input field.

CJK script

For CJK script input, several NetWeaver input fields are concatenated into the last multiline field.

If a descriptor is found on NetWeaver_House_Num2, the multiline is processed as follows (+ indicates that the fields are concatenated with no space between them):

```
NetWeaver_Street
+NetWeaver_House_Num1+NetWeaver_House_Num2+<Descriptor>NetWeaver_RoomNumber
+<Descriptor>NetWeaver_Floor+NetWeaver_Building
```

If no descriptor is found on NetWeaver_House_Num2, the multiline is processed as follows:

```
NetWeaver_Street+NetWeaver_House_Num1+NetWeaver_RoomNumber<Descriptor>
+NetWeaver_Floor<Descriptor>+NetWeaver_Building NetWeaver_House_Num2
```

Related Information

Input fields [page 1227]

3.5.4.10.2.4 Output fields

The following are Data Services output fields that can be used for the Global Address Cleanse transform. The Field_AddrClass and Field_Class values are available in the Global Address Cleanse transform's Transform Configuration Editor on the Output Best Practices tab for each field.

The table also shows that each field is available based on the engine(s) that you enable:

- Canada (C)
- Global Address (G)
- USA (U)

Output field name	Description	Engine
Additional_Info1	*Austria*: Includes the PAC code of the currently valid address when you choose to preserve the alias address on output. *Belgium*: Includes the NIS code. *Canada*: The official 13-character abbreviation of the city name, or the full spelling if the city name is less than 13 characters (including spaces). *France*: Includes the INSEE code. *Germany*: Includes a portion of the German freightcode (Frachtleitcode). *Liechtenstein*: Includes the postal service district (Botenbezirke) when it is available in the data. *Poland*: Includes the district name (powiat). *Spain*: Includes the INE 91 section code. *Switzerland*: Includes the postal service district (Botenbezirke) when it is available in the data.	C, G
Additional_Info2	*Austria*: Includes the City ID (OKZ). *Canada*: The official 18-character abbreviation of the city name, or the full spelling if the city name is less than 18 characters (including spaces). *Germany*: Includes the District Code. *Liechtenstein*: Additional postcode. *Poland*: Includes the community name (gmina).	C, G

Output field name	Description	Engine
	Spain: Includes the INE Street code. *Switzerland*: Additional postcode.	
Additional_Info3	*Austria*: Includes the Pusher-Leitcode (parcel). *Germany*: Includes the German City ID (ALORT). *Spain*: Includes the INE Town code.	G
Additional_Info4	*Austria*: Includes the Pusher-Leitcode (letter). *Germany*: Includes the German street name ID (StrSchl).	G
Additional_Info5	*Austria*: Includes the SKZ Street Code (7-digit). *Germany*: Includes the discount code for the freight-code.	G
Additional_Info6	*Austria*: Includes the corner-house identification (1-digit). The value for a corner house is *1*.	G
Additional_Info7-8	Reserved for future use.	All engines
Address_Line_Remainder1-4	Extraneous data found in the address line, which either cannot be identified or does not belong in a standardized address. USA 1-2: Complete secondary non-postal address (for example, Apt. 10, Ste 500, Box 34, Rm 7, 5th Flr).	All engines
Address_Type	A one-character code that represents the type of address identified: *P*: Postal *S*: Street *X*: Unknown	All engines
Area_Name1	An industrial area such as RIICO INDUSTRIAL AREA.	G
Assignment_Info	Indicates whether a record is valid, invalid, or corrected, based on the status and information codes. *C*: Corrected *I*: Invalid *V*: Valid	All engines
Assignment_Level	The level to which this transform matched the address to the data in the reference files (directories):	All engines

Output field name	Description	Engine
	C: Country *L1*: Locality1 *L2*: Locality2 *L3*: Locality3 *L4*: Locality4 *PN*: Primary name *PR*: Primary range *R*: Region *S*: Secondary *X*: Unknown, or the address was unassigned	
Assignment_Type	A one- or two-character code that represents the type of address. Engine support varies; see each code listing for supported engines. *BN*: Building name (Canada, Global Address) *F*: Firm (Canada, Global Address, USA) *G*: General delivery (Canada, Global Address, USA) *H*: High-rise building (Canada, USA) *HB*: House Boat (Global Address) *L*: LOT (Global Address) *M*: Military (Canada, USA) *R*: Rural (Canada, USA) *P*: Postal (Canada, Global Address, USA) *PI*: Point of reference (Global Address) *PR*: Poste Restante (Global Address) *PS*: Packstation or Paketbox (Global Address) *RP*: Postal Served by Route (Global Address) *S*: Street (Canada, Global Address, USA) *SR*: Street served by route (Canada, Global Address) *U*: Uninhabited (Global Address) *W*: Caravan (Global Address)	All engines

Output field name	Description	Engine
	X: Unknown or the address was unassigned (Canada, Global Address, USA)	
Block_Description	Block description such as "Block."	G
Block_Number	Block number.	G
Building_Name1	The building name for the address, which in some countries is used in place of the primary number. For example, in the U.K. an address may be “White House, High Street,” where “White House” is the building name instead of a primary number in an address such as “100 High Street.”	G
Building_Name2	The building name for the address, which in some countries is used in place of the primary number.	G
Cert_Valid	Indicates a valid certification.	All engines
Country	The ISO country code or the country name of the input record. The parsed value of this component is the country data found in the input record.	All engines
Country_Name	Fully-spelled country name in the languages specified in the Output_Country_Language option.	All engines
County_Name	Fully spelled county name. *USA*: County information is not included on mail pieces.	U
Delivery_Installation_Name	The delivery installation city name, which is usually the same as the city name and (if it is the same) omitted from the address line. *Canada*: If the delivery installation name is different than the locality name, the delivery installation name is output to the secondary address fields. *Japan*: Returns the post office name.	C, G
Delivery_Installation_Qualifier	Delivery Installation qualifier (for example, “Main” in “RR 2 Vancouver Stn Main”).	C
Delivery_Installation_Type	The delivery installation type. *English*: *PO*: Post Office *RPO*: Retail Post Outlet *STN*: Station	C

Output field name	Description	Engine
	LCD: Letter Carrier Depot *CMC*: Community Mail Center *CDO*: Commercial Dealership Outlet *French*: *BDP*: Bureau de Poste *CSP*: Comptoir Service Postal *SUCC*: Succursale. *PDF*: Poste de Facteurs *CPC*: Centre Postal Communautaire *CC*: Concession Commerciale	
Delivery_Point	*Australia*: Eight-digit delivery point identifier. This is the primary component needed to generate a barcode. This component is not printed on mail pieces. *Austria*: Includes the PAC code, which is a unique identifier assigned by the Austrian postal authority. *New Zealand*: A seven-character code that represents the delivery-point identifier. *United Kingdom*: A two-character code that represents the delivery-point suffix.	G
Engine_Name	The name of the engine that was selected to process the record.	All engines
Error	Specifies the error status generated as the result of looking up the current record and performing suggestion processing. Possible output values are 0- 5. *0* No suggestion selection error. *1* Blank suggestion selection/entry. *2* Invalid suggestion selection. *3* Invalid primary range. *4* Invalid floor range. *5* Invalid unit range.	All engines
Extra1-12	Any non-address data found in the address block. Available only if the input data was presented through multiline fields.	All engines

Output field name	Description	Engine
Firm	The firm name for the address. Identification of firm name data in a multiline format may be inconsistent depending upon the level of firm data available in the postal directories for each engine. To avoid inconsistent identification of firm data, use the discrete Firm field when you process multiline data. *Canada and USA*: The firm name is taken from the postal directory if found; otherwise, it's taken from the input record. Be aware that the postal directory might contain some unusual or shortened spellings that you may or may not find suitable for printing on mail pieces. If you prefer to retain your own firm data, retrieve the parsed component. *Global Address*: If the firm name is available on input, the Global Address engine returns the firm name.	All engines
Floor_Description	The level description, such as "Floor." *Japan*: The level description, such as kai.	G
Floor_Number	The level number or information.	G
Floor_Qualifier	Additional word that precedes or follows the floor information.	G
Full_Address	The complete address line, including secondary address, and dual address (street and postal).	All engines
Info_Code	If the address is not fully assigned, displays a four-character code that describes why the address could not be assigned. If the address is fully assigned, the field is blank. For more information, see *Information codes (Global Address Cleanse)* [page 1409].	All engines
ISO_Country_Code_2Char	The two-character ISO code that identifies a country, for example, DE is Germany.	All engines
ISO_Country_Code_3Char	The ISO-3166 three-character code that identifies a country, for example, DEU is Germany.	All engines
ISO_Country_Code_3Digit	The three-digit ISO code that identifies a country, for example, 276 is Germany.	All engines
ISO_Script_Code	The four-character script code to use for an identified country, such as LATN or KANA.	All engines

Output field name	Description	Engine
Language	The two-character ISO language code that represents the language of the address.	All engines
Lastline	The locality (Locality1–Locality4 if available), region, and postal code together in one component. The region is only included when it is required for select countries.	All engines
Lastline_Remainder1-4	Unused lastline remainder data.	G
Locality1_Addition	Additional locality information.	G
Locality1_Alternate	Preserves the input locality if it is recognized by the postal authority as a locality name for this address. Misspellings are corrected.	C, U
Locality_Code	Used in some countries to distinguish sections of a large locality. For example, in France they are called arrondissements.	G
Locality1_Description	Locality1 descriptor. *Japan*: Locality1 descriptor. For example, shi, shima, and so on. *China*: Locality1 descriptor. For example, 市(Shi).	G
Locality2_Description	Description of a subdivision of Locality1.	G
Locality3_Description	Description of a subdivision of Locality2.	G
Locality4_Description	Description of a subdivision of Locality3.	G
Locality1_Full	Includes Locality1_Name, Locality_Code, Locality1_Description, and Locality1_Qualifier. It may include Locality1_Addition, depending on the standardization option settings of Locality Name Style and Include Locality Addition.	All engines
Locality2_Full	Includes Locality2_Name and Locality2_Description.	G, U
Locality3_Full	Includes Locality3_Name and Locality3_Description.	G
Locality4_Full	Includes Locality4_Name and Locality4_Description.	G
Locality1_Name	The city, town, locality, or suburb that is either the Locality1_Alternate or Locality1_Official, depending on the standardization option setting for Assign Locality.	All engines

Output field name	Description	Engine
	Japan: The city (shi), island (shima), ward (ku), county (gun), district (machi), or village (mura).	
Locality2_Name	Additional locality information. *USA*: Urbanization (Puerto Rican addresses only).	G, U
Locality3_Name	Additional locality information.	G
Locality4_Name	Additional locality information.	G
Locality1_Official	The locality name preferred by the postal authority.	All engines
Locality2_Official	The locality name preferred by the postal authority.	G
Locality3_Official	The locality name preferred by the postal authority.	G
Locality4_Official	The locality name preferred by the postal authority.	G
Locality1_Qualifier	Used by France for Cedex.	G
Locality2_Qualifier	Locality qualifier.	G
Locality3_Qualifier	Locality qualifier.	G
Locality4_Qualifier	Locality qualifier.	G
Match_Block_Number	The standardized form of Block_Number used in the Match transform during the comparison process. Block descriptions are not output. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency.	All engines
Match_Building_Name	The standardized form of Building_Name1 used in the Match transform during the comparison process. Builiding descriptions are output. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is se-	All engines

Output field name	Description	Engine
	lected. For CJK scripts, the field uses normal width standardization for consistency. ℹ Note For China, building indicators will be removed.	
Match_Floor_Number	The standardized form of Floor_Number used in the Match transform during the comparison process. Floor descriptions and qualifiers are not output. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency.	All engines
Match_Locality	The standardized form of locality used in the Match transform during the comparison process. The output is not affected by standardization settings. Locality1_Official is output when a locality or better level assignment is made; otherwise, the Locality1_Name is output. Locality codes, qualifiers, or descriptions are not output. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency. ℹ Note For China and Japan, Locality1–4_Official or Locality1–4_Name are output, if present. For all other countries, only Locality1_Official or Locality1_Name is output.	All engines
Match_Primary_Directional	The standardized form of Primary_Prefix1 and Primary_Postfix1 used in the Match transform during the comparison process. The output is not affected by standardization settings. The abbreviated form is	All engines

Output field name	Description	Engine
	output, if available. If a prefix and postfix are both present, they are separated by a space. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency.	
Match_Primary_Name	The standardized form of Primary_Name1 used in the Match transform during the comparison process. The output is not affected by standardization settings. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. Prefix, postfix, suffix, and type data is removed. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency. Note For Brazil, China, and Japan, Primary_Name1–4 are output, if present. For all other countries, only Primary_Name1 is output.	All engines
Match_Primary_Number	The standardized form of Primary_Number used in the Match transform during the comparison process. Only the Primary_Number and Primary_Number_Extra are output, not the Primary_Number_Description. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency.	All engines
Match_Primary_Type	The standardized form of Primary_Type1–4 used in the Match transform during the comparison process. The output is not affected by standardization set-	All engines

Output field name	Description	Engine
	tings. The abbreviated primary type is output, if available. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency. Note For Brazil, China, and Japan, Primary_Type1–4 are output, if present. For all other countries, only Primary_Type1 is output.	
Match_Stairwell_Name	The standardized form of Stairwell_Name used in the Match transform during the comparison process. Stairwell descriptions are not output. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency.	All engines
Match_Unit_Number	The standardized form of Unit_Number used in the Match transform during the comparison process. Unit descriptions and qualifiers are not output. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency.	All engines
Match_Wing_Name	The standardized form of Wing_Name used in the Match transform during the comparison process. Wing descriptions are not output. Data is output in uppercase, diacritical characters and apostrophes are removed, and other punctuation and multiple	All engines

Output field name	Description	Engine
	spaces are replaced with a single space. The field is available only as a Best component. Non-Latin scripts are transliterated for supported scripts if the Output Address Script option is selected. For CJK scripts, the field uses normal width standardization for consistency.	
Multiline1-12	A line that may contain any data. The type of data in this line may vary from record to record.	All engines
NetWeaver_Formatted_Postcode	The postcode in a format that SAP software requires.	All engines
PMB_Full	Contains private mailbox information.	All engines
PName_Secondary_Addr	Contains the full primary name (with no associated primary number) and the full secondary address.	All engines
Point_Of_Reference1-2	A well known place or easily visible location to help locate an address. For example, Opposite to Citibank ATM.	G
Post_Office_Name	The name or numeric representation for a post office, such as, "01" BP 1012.	G
Postcode_Description	A word that indicates a postal code, when available on input. For example: *Brazil*: CEP, which stands for Código de Endereçamento Postal, and is output as CEP 52041-970. *China*: 邮编 *Japan*: 〒	G
Postcode_Full	*Australia*: Complete four-digit postal code. *Canada*: Complete six-character postal code (FSA + LDU). *Global Address*: Complete postal code. *USA*: The full ZIP Code with a hyphen (10 characters). *Japan*: The seven-digit postal code.	All engines
Postcode_In_NetWeaver_Supported_Format	Indicates whether the NetWeaver_Formatted_Postcode output field is populated.	All engines
Postcode_Prefix	The postcode prefix that is used by some European countries. For example, many countries use the same postal code format of four or five digits. You can prefix the numeric postal code with a country code to	G

Output field name	Description	Engine
	avoid confusion when sending mail to or from the European country. The codes used are generally based on License plate codes (D for Germany or F for France) rather than ISO codes.	
Postcode1	*Australia*: Four-digit postcode. *Canada*: First three characters (FSA) of the postal code. *Global Address*: Postal code. *USA*: Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4). *Japan*: The first three digits of the postal code.	All engines
Postcode2	The secondary postal code. *Canada*: The last three characters (LDU) of the postal code. *USA*: The four-digit ZIP Code, which on a mail piece, this code follows the primary postal code with a hyphen placed between (for example, 54601-1234). *Japan*: Contains the last four digits of the postal code.	All engines
Primary_Address	Primary address line, such as the street address or post office box. Does not include secondary address information such as apartment. *Japan*: The full block data.	All engines
Primary_Delivery_Mode	The delivery mode for a street served by route type address (Rural Route).	C
Primary_Delivery_Number	The delivery number for a street served by route type address (Rural Route).	C
Primary_Name1	The street name description (typically a street name or box description). *Japan*: Block (chome, kumi, Hokkaido go), sub-block (banchi, gaiku, tochi kukaku). The Post office name description (yuubinnkyoku or siten).	All engines
Primary_Name2	Second street and name description, typically a street name or box description. *Japan*: Additional block and sub-block information.	G

Output field name	Description	Engine
Primary_Name3	The street name, delivery mode, and so on. *Japan*: Additional block and sub-block information.	G
Primary_Name4	The street name, delivery mode, and so on. *Japan*: Additional block and sub-block information.	G
Primary_Name_Full1	The primary name, primary type, primary prefix, and primary postfix.	All engines
Primary_Name_Full2	The primary name2, primary type2, primary prefix2, and primary postfix2.	G
Primary_Name_Full3-4	The primary name and primary type.	G
Primary_Number_Description	A description preceding the primary number. For example, KM (Kilometer) or Blk. *Japan*: The postal number identifier 号 (go) or house number description 号 (go). *China*: The description after street number. For example, 号(hao).	G
Primary_Number	The premise number, rural route number, or PO Box number. In some cases it may include a range.	All engines
Primary_Number_Extra	Data found near the parsed primary number, which in most cases cannot be identified or does not belong in a standardized address. *Japan*: The postal box identifier.	G
Primary_Number_Full	The primary number, primary number description, and primary number extra.	All engines
Primary_Postfix1	Abbreviated or non abbreviated directional (for example, N, South, NW, SE) that follows a street name. Abbreviated or non abbreviated is based on the standardization setting for Directional Style. *Japan*: Directional that follows block or sub-block.	All engines
Primary_Postfix2	Abbreviated or non abbreviated directional (for example, N, South, NW, SE) that follows a street name. Abbreviated or non abbreviated is based on the standardization setting for Directional Style. *Japan*: Directional that follows block or sub-block.	G
Primary_Prefix1	Abbreviated or non abbreviated directional (N, South, NW, SE) that precedes a street name. Abbreviated or	G, U

Output field name	Description	Engine
	non abbreviated is based on the standardization setting for Directional Style. *Japan*: Directional that precedes block or sub-block.	
Primary_Prefix2	Abbreviated or non abbreviated directional (N, South, NW, SE) that precedes a street name. Abbreviated or non abbreviated is based on the standardization setting for Directional Style. *Japan*: Directional that precedes block or sub-block.	G
Primary_Secondary_Address	The primary address and secondary address in one component.	All engines
Primary_Type1	The type of primary name (some examples are rue, strasse, street, Ave, or Pl).	All engines
Primary_Type2-4	The type of primary name (some examples are rue, strasse, street, Ave, or Pl).	G
Quality_Code	Displays a two-character code that provides additional information about the quality of the address. The quality of the address depends on the input data, the processing engine, country, information code, and status code (if an information code is not generated). For more information, see *Quality codes (Global Address Cleanse)* [page 1416].	All engines
Region1	Either the Region1_Name or Region1_Symbol based on the standardization option Region Style.	All engines
Region1_Code	The region code, which may be the ISO region code.	All engines
Region1_Description	The Region1 description.	G
Region1_Full	Includes Region1 and Region1_Description.	All engines
Region1_Name	The fully spelled out Region1 name.	All engines
Region1_Symbol	An abbreviation of the Region1 name.	All engines
Region2	Either the Region2_Name or Region2_Symbol based on the standardization option Region Style.	G
Region2_Code	The region code, which may be the ISO region code.	All engines
Region2_Description	The Region2 description.	G

Output field name	Description	Engine
Region2_Full	Includes Region2 and Region2_Description.	G
Region2_Name	The fully spelled out Region2 name.	G
Region2_Symbol	An abbreviation of the Region2 name.	G
Remainder_Full	Contains all remainder information, including Address_Line_Remainder1-4 and Lastline_Remainder1-4.	All engines
Secondary_Address	The floor, unit, stairwell, or wing data on one line.	All engines
Single_Address	The full address and last line in one component.	All engines
Stairwell_Description	Entrance or stairwell identifier for a building, such as, Entrada "1."	G
Stairwell_Name	The name or number of an entrance or stairwell for a building, such as, Entrada "1."	G
Status	Specifies the suggestion status generated as the result of looking up the current record and performing suggestion processing. *A*: Primary address-line suggestions available. *AM*: Follow up primary address-line suggestions available. *F*: Floor range is invalid. *L*: Lastline suggestions available. *N*: No suggestions available. *R*: Primary range is invalid. *S*: Unit range is invalid. *U*: Secondary address-line suggestions available. *UM*: Follow up secondary address-line suggestions available.	All engines
Status_Code	Displays a six-character code that always starts with an S. This code explains what parts of the address changed during processing. For more information, see *Status codes (Global Address Cleanse)* [page 1412].	All engines
Unit_Description	The unit description, such as “Apartment” or “Flat.” *Japan*: The unit description, such as gousitsu.	All engines

Output field name	Description	Engine
Unit_Number	The unit number, such as 100 in “Apartment 100.”	All engines
Unit_Qualifier	Additional word that precedes or follows the unit information.	G
Wing_Description	Identifies a wing within a building, such as, West "Wing."	G
Wing_Name	The name or number of a wing within a building, such as "West" Wing.	G

Related Information

Information codes (Global Address Cleanse) [page 1409]
Quality codes (Global Address Cleanse) [page 1416]
Status codes (Global Address Cleanse) [page 1412]

3.5.4.10.2.5 Global Address Cleanse Suggestion List fields

The Global Address Cleanse transform's Suggestion List option requires that you map fields on input and output.

- *Suggestion List Input Fields* [page 1249]
- *Suggestion List Output Fields* [page 1250]

3.5.4.10.2.5.1 Suggestion List Input Fields

The Global Address Cleanse transform's Suggestion List option supports all Global Address Cleanse input fields in addition to the suggestion reply fields.

Field	Description
Suggestion_Reply1–6	Contains the reply when more information is needed to complete the query. Each of these fields also contain the reply if a selection from a list needs to be made. Possible types of generated suggestion lists are: ● Lastline ● Primary Address ● Follow-up Primary Address ● Secondary Address ● Follow-up Secondary Address

3.5.4.10.2.5.2 Suggestion List Output Fields

The following are fields that you can use for the Global Address Cleanse transform's Suggestion List option. The fields are listed alphabetically.

Field	Description
Building_Name	The building name for the address, which in some countries is used in place of the primary number. For example, in the U.K. an address may be “White House, High Street,” where “White House” is the building name instead of a primary number in an address such as “100 High Street.”
Delivery_Installation_Name	The delivery installation city name, which in some cases is the same as the city name and (if it is the same) omitted from the address line.
Delivery_Installation_Qualifier	Delivery Installation qualifier (for example, “Main” in “RR 2 Vancouver Stn Main”).
Delivery_Installation_Type	The delivery installation type. *English*: • *PO*: Post Office. • *RPO*: Retail Post Outlet. • *STN*: Station. • *LCD*: Letter Carrier Depot. • *CMC*: Community Mail Center. • *CDO*: Commercial Dealership Outlet. *French*: • *BDP*: Bureau de Poste. • *CSP*: Comptoir Service Postal. • *SUCC*: Succursale. • *PDF*: Poste de Facteurs. • *CPC*: Centre Postal Communautaire. • *CC*: Concession Commerciale.
Firm	The firm name for the address.
Floor_Description	The level description, such as “Floor.”
Floor_Number_High Floor_Number_Low	If the floor number is a range such as 20-22, LOW contains “20” and HIGH contains “22.” If the floor number is not a range, both fields contain the floor number (for example, “20” and “20”).
Locality1 Locality2 Locality3 Locality4	The city, town or suburb and any additional related information.

Field	Description
Locality1_Official Locality2_Official Locality3_Official Locality4_Official	The locality name preferred by the postal authority.
Postcode	The postal code. *USA*: The five-digit ZIP Code and ZIP+4.
Postcode1	*Australia*: Four-digit postcode. *Canada*: First three characters (FSA) of the postal code. *Global*: Postal code. *USA*: Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4).
Postcode2	The secondary postal code. *Canada*: The last three characters (LDU) of the postal code. *USA*: The four-digit ZIP Code, which on a mail piece, this code follows the primary postal code with a hyphen placed between (for example, 54601-1234).
Primary_Name1	The street name description (typically a street name or box description).
Primary_Name2	Second street name and description, typically a street name or box description.
Primary_Name3 Primary_Name4	The street name, delivery mode, and so on.
Primary_Name_Full1 Primary_Name_Full2	The primary name, primary type, primary prefix, and primary postfix.
Primary_Name_Full3 Primary_Name_Full4	The primary name and primary type.
Primary_Number_Description	A description preceding the primary number. For example, KM (Kilometer) or Blk.
Primary_Number_Extra	Data found near the parsed primary number, which in most cases cannot be identified or does not belong in a standardized address.
Primary_Number_Full	The primary number, primary number description, and primary number extra.
Primary_Number_High Primary_Number_Low	If the house number is a range such as 100-102, LOW contains “100” and HIGH contains “102.” If the house

Field	Description
	number is not a range, both fields contain the house number (for example, “100” and “100”).
Primary_Postfix1 Primary_Postfix2	Abbreviated or non-abbreviated directional (for example, N, South, NW, SE) that follows a street name.
Primary_Prefix1 Primary_Prefix2	Abbreviated or non-abbreviated directional (N, South, NW, SE) that precedes a street name.
Primary_Side_Indicator	Indicates if even, odd, or both values are valid. This applies to streets and PO Boxes. *E*: The record covers the even-numbered value. *O*: The record covers the odd-numbered value. *B*: The record covers both the even- and odd-numbered values.
Primary_Type1 Primary_Type2 Primary_Type3 Primary_Type4	The type of primary name (rue, strasse, street, Ave, or Pl).
Region1	Returns the state, province, or region.
Secondary_Side_Indicator	Indicates if even, odd, or both values are valid. This applies to floors and units. *E*: The secondary record covers the even-numbered value. *O*: The secondary record covers the odd-numbered value. *B*: The secondary record covers both the even- and odd-numbered values.
Selection	A unique index number that identifies this suggestion from the others in the returned list. The suggestion “selection” number ranges from 1 to the number of suggestion selections in the suggestion list.
Stairwell_Description	Entrance or stairwell identifier for a building, such as, "Entrada" 1.
Stairwell_Name	The name or number of an entrance or stairwell for a building, such as Entrada "1."
Unit_Description	The unit description, such as “Apartment” or “Flat.”
Unit_Number_High Unit_Number_Low	If the unit number is a range such as 20-22, LOW contains “20” and HIGH contains “22.” If the unit number is not a range, both fields contain the unit number (for example, “20” and “20”).

3.5.4.10.3 Global Address Cleanse sample configurations

For specialized processes like cleansing address data in Australia or Brazil, Data Services has Global Address Cleanse sample transform configurations that you can include in your data flows. Find the sample transform configurations in Data Services Object Library under Global_Address_Cleanse.

> Note
>
> Sample configurations include all required options except input fields. All sample configurations display in the designer as Global_Address_Cleanse.

Sample transform name	Description
Australia_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Australia.
Brazil_ AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Brazil.
Canada_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Canada.
China_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in China.
Europe_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in multiple European countries.
France_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in France.
Germany_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Germany.
Global_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse Latin script address data in any supported country.
GlobalSuggestions_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse Latin-1 address data in any supported country using the Suggestion List feature.
Greece_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Greece when the address data consists of Greek Data
Italy_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Italy.
Japan_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Japan when the address data consists of Japanese Kanji, Katakana, and Hiragana.

Sample transform name	Description
Portugal_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Portugal.
Spain_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Spain.
UK_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in the United Kingdom.
USA_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in the United States.
USASuggestions_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in the United States using the Suggestion List feature.

3.5.4.11 Global Suggestion List

The Global Suggestion List transform query addresses with minimal data, and it can offer suggestions for possible matches. It is a beneficial research tool for managing unassigned addresses from a batch process.

Global Suggestion List functionality is designed to be integrated into your own custom applications via the Web Service. If you are a programmer looking for details about how to integrate Global Suggestion List functionality, see "Integrate Global Suggestion List functionality" in the "Detailed concepts for integrators" section of your *Data Services Integrator Guide*.

The Global Suggestion List transform requires the two character ISO country code on input. Therefore, you may want to place a transform, such as the Country ID transform, that will output the ISO_Country_Code_2Char field before the Global Suggestion List transform.

The Global Suggestion List transform is available for use with the Canada, Global Address, and USA engines.

> Note
>
> No certification with Global Suggestion List: If you use the Canada engine, USA engine, or Global Address engine for Australia and New Zealand, you cannot certify your mailing for SERP, CASS, AMAS, or New Zealand certification.

> Note
>
> This option does not support processing of Japanese or Chinese address data.

Related Information

Global Suggestion List option groups [page 1255]
Designer Guide: Cleanse your address data transactionally [page 593]

3.5.4.11.1 Content objects

Transform configurations

A transform configuration is a transform with preconfigured input fields, output fields, and options that can be used in multiple data flows. These are useful if you repeatedly use a transform with specific options and input and output fields.

When Data Services is installed, read-only transform configurations are provided for the Data Quality transforms.

You can use transform configurations in your data flows or as an example of a typical transform. After you place an instance of the transform configuration in a data flow, you can override these preset defaults. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one.

Sample blueprints and other objects

We have created Data Quality blueprints and other content objects to help you set up Data Services jobs. We've identified a number of common scenarios that you are likely to perform with Data Services. For each scenario, we've included a blueprint that is already set up to solve the business problem in that scenario.

Related Information

Downloading blueprints and other content objects [page 1128]
Transform configurations [page 1123]

3.5.4.11.2 Global Suggestion List option groups

The Global Suggestion List transform includes the following option groups.

> Note
>
> The Global Suggestion List transform does not support Chinese and Japanese addresses.

Option group	Description
Common: Run as Separate Process	Splits the transform into a separate process.
Engines	Sets the engines for processing. The default value for each country engine is *Yes*. Select *No* to disable the country engine. Countries include: ● Canada ● Global Address ● USA
Canada	Sets the directory path for your address cleanse reference files.
Global Address	Sets the directory path for your address cleanse reference files.
USA	Sets the directory path for your address cleanse reference files.
Reference Files	Specifies the location of your reference files. You can use a substitution variable.
Options	See *Options (Global Suggestion List)* [page 1257]
Suggestion List	See *Suggestion List (Global Suggestion List)* [page 1257].

3.5.4.11.3 Engines (Global Suggestion List)

The Engines option group allows you to enable or disable individual engines of the Global Suggestion List transform.

> Note
>
> The Suggestion List option does not support Chinese and Japanese addresses.

Option	Description
Canada	Specifies if the engine is enabled or disabled for suggestion List processing. Choose one of the following: *Yes:* Enables the engine. *No:* Disables the engine.
Global Address	Specifies if the engine is enabled or disabled for suggestion List processing. Choose one of the following: *Yes:* Enables the engine. *No:* Disables the engine.
USA	Specifies if the engine is enabled or disabled for suggestion List processing. Choose one of the following: *Yes:* Enables the engine.

Option	Description
	No: Disables the engine.

3.5.4.11.4 Options (Global Suggestion List)

This option group contains all of the settings that you need to define when you process data with the Global Suggestion List transform.

> Note
>
> The Global Suggestion List transform does not support Chinese and Japanese addresses.

Option	Description
Default Country	Specifies a country to use if the input field is not defined or if the Country input field is blank. Enter a valid two-character country code, or enter *None* if you do not want to use a default country.
Enter Firm Data	Specifies whether to request firm data when the selected suggestion does not have data available. *Yes:* Requests firm data. *No:* Does not request firm data.
Return Single Item Lists	Specifies whether the transform should return suggestion lists that have only one item. *Yes:* Returns suggestion lists that only have one item. *No:* If a suggestion list has only one item, then the single item is automatically selected from the suggestion list and processing continues.

3.5.4.11.5 Suggestion List (Global Suggestion List)

Use this option group to configure options for how suggestion lists are output, as well as set up the fields where the suggestion information is posted.

Yes: Outputs the component.

No: Does not output the component.

Table 160: Suggestion List options

Option/Option group	Description
Output Style	Specifies the format for the output suggestion list data. *Delimited:* Outputs the suggestion list data in a delimited text format, with the delimiters specified in the Delimiter and Field Delimiter options. *XML:* Outputs the suggestion list data as hierarchical XML. If you integrate suggestion lists via the web service, you are likely to use this option. You can then use the XML tools you own to parse through the suggestion list data.
Delimiter	Specifies a character to use to separate each suggestion in a suggestion list. This value is considered only if the Style option is set to Delimited. This value can be any character or string. Common delimiters include a pipe symbol (|), or a string of multiple asterisks (***). This value should differ from the Field Delimiter value.
Field Delimiter	Specifies a character to use to separate each field in a single suggestion. This value is considered only if the Style option is set to Delimited. Each suggestion can be made up of one or more fields. If you choose to retrieve multiple fields per suggestion, those fields are separated with the character specified here. This value can be any character or string. The default value is a pipe symbol (|). This value should differ from the Delimiter value.

Table 161: Lastline Components

Option/Option group	Description
Selection	Returns a unique index number that identifies this suggestion from the others in the returned list.
Locality1-3	Returns the city, town, or suburb. Additional locality information goes in Locality2.
City Addition	Returns unofficial city information that is associated with the locality. For example, there are two German cities named Frankfurt. The larger city of Frankfurt is called Frankfurt am Main and the smaller city is called Frankfurt (Oder). Locality1 would have Frankfurt for these two records, and City Addition would list (Oder) and am Main.
Region1	Returns the state, province, territory, or region of the address.
Postcode1	Returns the postal code or five-digit ZIP Code (USA).
Primary Names Available	Indicates whether or not street data is available for a locality: *Y*: Yes, there are streets. *N*: No, there are not streets.

Table 162: Primary Name Components

Option/Option group	Description
Selection	Returns a unique index number that identifies this suggestion from the others in the returned list.
Primary Name1 Primary Name2	Returns the street description. For example, Primary Name1 may return "Marina" and Primary Name2 may return "The Slipway."
Locality1-3	Returns the city, town, or suburb. Additional locality information goes in Locality2.
City Addition	Returns unofficial city information that is associated with the locality. For example, there are two German cities named Frankfurt. The larger city of Frankfurt is called Frankfurt am Main and the smaller city is called Frankfurt (Oder). Locality1 would have Frankfurt for these two records, and City Addition would list (Oder) and am Main.
Postcode1	Returns the postal code.

Table 163: Address Components

Option/Option group	Description
Selection	Returns a unique index number that identifies this suggestion from the others in the returned list.
Primary Side Indicator	Indicates if even, odd, or both values are valid. This applies to Street and PO box. *E*: The record covers the even-numbered values. *O*: The record covers the odd-numbered values. *B*: The record covers both the even- and odd-numbered values.
Firm	Returns the name of a firm, company, or organization.
Multiline1-6	Returns individual formatted address lines. This will not include country information, as it will be output in a separate field.
Postcode1	Returns the postal code.

3.5.4.11.6 Input fields

The following are input fields that you can use in the input mapping for the Global Suggestion List transform. The fields are listed alphabetically.

Note

The Global Suggestion List transform uses all fields provided on input to select a match and will return an error stating "no results" if the fields you include do not exist in the data. If you do not own address-level data for a country, do not include address-level fields in the input mapping. Also try removing address-level fields when address-level data is available, but Address Lists states there are no results. This allows you to view Locality-level data for the address.

ℹ Note

Global Suggestion List does not support Chinese and Japanese addresses.

Field	Description
Country	Specifies the country to look up in the query. This field must be the two-character country ISO code, not a country name. If your data does not contain the country code, place a transform that generates the country code field, such as the Country ID transform or Global Address Cleanse transform before the Global Suggestion List transform. If this field is blank, the transform uses the country found in the Default Country option.
Locality1	Specifies the city, town, or suburb.
Postcode	Specifies the postal code to look up.
Primary_Name1	Specifies the primary street name to look up. For example, in "255 Main St" the primary name is "Main."
Primary_Number	Specifies the primary number to look up. For example, in "255 Main St." the primary number is "255."
Reply1-5	Contains the reply when more information is needed to complete the query. Each of these fields also contain the reply if a selection from a list needs to be made. Possible types of generated suggestion lists are lastline, primary name, and address. ➜ Remember These fields must be added in ascending order; that is, you should populate Reply1 first, Reply2 second, and so on.
Start_Selection	Specifies the starting list number. If left blank, the default value is 1.
Script_Default	Sets the output script. This option is valid for Greece. If the input data is non-numeric, then the script type is determined by the script of the input data. For example, when the input data contains only numeric data for Greece and the option is set to 2, then the generated suggestion lists are output in Greek script, respectively. In the same situation except where the option is set to 1, then the generated suggestion lists are output in Latin script.

3.5.4.11.7 Output fields

The following are fields that you can use in the output mapping for the Global Suggestion List transform. The fields are listed alphabetically.

Note

Global Suggestion List does not support Chinese and Japanese addresses.

Field	Description
Country_Name	Returns the fully-spelled country name, in English.
Data_Type	Returns a single-character code that indicates what type of additional data is needed to complete processing. Possible output values are: *N:* No additional data is needed. *A:* Primary address data is needed. *F:* Firm data is needed. *R:* Primary range data is needed. *S:* Secondary range data is needed.
Error	Posts the error status generated as the result of looking up the current record and performing query processing. Possible output values are: *0:* There were no query errors. *1:* There was a system error while performing the query. *2:* The suggestion selection was invalid. For example, a selection of 8 is made and there are only 5 entries.
Firm	Specifies the firm name for the address.
Multiline1-6	Returns a line that may contain any data. The type of data in this line may vary from record to record. If you want to output the postal code, you cannot use one of these Multiline output fields. You must use the Postcode1 field.
Postcode1	Returns the postcode. *Canada and Global Address:* Postal code. *USA:* Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4).
Status	Returns a code indicating query status generated as the result of processing the input record and performing Global Suggestion List processing. Possible output values are: *C:* Querying is complete. *D:* More data is needed. *E:* There was an error. *P:* A suggestion list was generated.

Field	Description
Suggestion_Count	Returns the number of individual suggestion selections generated as the result of querying the current record. A nonnegative value is output. If the input record did not generate a suggestion list, this field contains a value of 0. Your application developer uses this field to know how many suggestion selections must be displayed to users of your custom application.
Suggestion_List	Contains the list of suggestions based on the Suggestion List Option settings that you set in the Global Suggestion List transform.
Suggestion_Type	Returns a code indicating what type of suggestion list was generated. Possible output values are: *N:* No suggestion list was generated. *A:* An address suggestion list was generated. *L:* A lastline suggestion list was generated. *S:* A secondary suggestion list was generated.
System_Error_ Description	Posts the current Global Suggestion List system error as a descriptive string.
System_Error_ Number	Posts the current system error as a number. The values are: *0:* No error. *1:* Invalid postcode. *2:* Invalid street. *3:* Invalid town. *4:* More information needed. *5:* Street information needed. *6:* No input given. *7:* Postcode numeric. *8:* Town needed. *9:* Town or postcode needed. *10:* No street information available. *11:* Country blank. *12:* Invalid country. *13*: No results. *14:* Address needed. *15:* Premise needed. *16:* Firm needed.

Field	Description
Warning	Posts the warning status generated as the result of looking up the current record and performing query processing. The values are: *0:* There were no query warnings. *1:* An incomplete suggestion list was generated. Generally, the maximum number of elements that can be placed in a suggestion list is 200. *2:* An invalid premise was entered. *3:* An invalid unit was entered.

3.5.4.12 Match

The Match transform is responsible for performing matching based on the business rules you define. The transform then sends matching and unique records on to the next transform in the data flow.

For best results, the data in which you are attempting to find matches should be cleansed. Therefore, you may need to include other Data Quality transforms before the Match transform.

Match concepts

This section describes the Match transform, how it fits into a data flow, and the options you can set to conform to your business rules. The Match transform is only one tool, albeit the most important one, for you to use in your matching strategy. For more information about matching concepts and other transforms you can use to achieve the results you are looking for, see the Match section of the *Designer Guide*.

Related Information

Match transform tab [page 1264]
Group forming [page 1269]
Match level options [page 1276]
Post-match processing [page 1295]

3.5.4.12.1 Content objects

Transform configurations

A transform configuration is a transform with preconfigured input fields, output fields, and options that can be used in multiple data flows. These are useful if you repeatedly use a transform with specific options and input and output fields.

When Data Services is installed, read-only transform configurations are provided for the Data Quality transforms.

You can use transform configurations in your data flows or as an example of a typical transform. After you place an instance of the transform configuration in a data flow, you can override these preset defaults. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one.

Sample blueprints and other objects

We have created Data Quality blueprints and other content objects to help you set up Data Services jobs. We've identified a number of common scenarios that you are likely to perform with Data Services. For each scenario, we've included a blueprint that is already set up to solve the business problem in that scenario.

Related Information

Downloading blueprints and other content objects [page 1128]
Transform configurations [page 1123]

3.5.4.12.2 Match transform options

3.5.4.12.2.1 Match transform tab

These options control the Match transform itself. Here you control whether to generate report data, which match engine will be processing data, and so on.

Option	Description
Perform matching	Select to add the ability to compare records and other match-related operations. This option is not selected only in the Base_Match transform configuration. For all other configurations, this option is selected.

Option	Description
Match set name	Enter a name for this match set. Each Match transform in your data flow represents a match set. This option is already populated with name you chose if you used the Match wizard to generated this transform. Be sure that this name is unique within the data flow (it does not match the name of another match set). This name is used in the match reports to differentiate data processed by one match set versus another.
Match engine	Specifies the match engine to use, based on the type of data you will be processing. If you use the Multinational strategy in the Match wizard, this option is set to Latin1 for all match sets. *Chinese:* Specifies that the Match transform will be processing Chinese data in Chinese script. *Japanese:* Specifies that the Match transform will be processing Japanese data in Japanese script. *Korean:* Specifies that the Match transform will be processing Korean data in Korean script. *Latin1:* Specifies that the Match transform will be processing Latin1 data. In general, this is the data used throughout the Americas, Western Europe, Oceania, and much of Africa. *Other_Non_Latin1:* Specifies that the Match transform will be processing non-Latin1 data, other than Chinese, Japanese, Korean, and Taiwanese, such as Russian, Greek, Hebrew, Arabic, and others. *Taiwanese:* Specifies that the Match transform will be processing Taiwanese data in Taiwanese script. For optimum accuracy and performance, be sure that you have filtered your multinational data to separate match transforms with the appropriate match engine selected. The Match wizard can do this for you easily, if you select the Multinational strategy. By default, the Latin1 match engine is used. If you attempt to process non-latin1 data with the Latin1 engine, the results are unpredictable.
Generate report data	Specifies whether to generate report data for this transform. This option is available in every transform that generates report data. *Yes:* Generates report data for this transform. *No:* Turns off report data generation. If you do not need to generate reports (during testing of data flows, for example), you should set this option to No. This will improve match performance.
Logical source field	Specifies the field that contains the ID for the logical source.
Physical source field	Specifies the field that contains the ID for the physical source (Reader).

Option	Description
Run as a separate process	*Yes:* Splits the transform into a separate process. *No:* Keeps the transform in same process as the rest of the data flow.

Related Information

Performance Optimization Guide: Run as a separate process option [page 2153]

3.5.4.12.2.2 Match transform options: Data Salvage tab

Option	Description
Match set name	Enter a name for this match set. Each Match transform in your data flow represents a match set. This option is already populated with name you chose if you used the Match wizard to generated this transform. Be sure that this name is unique within the data flow (it does not match the name of another match set). This name is used in the match reports to differentiate data processed by one match set versus another.
Enable data salvage	Select to perform data salvaging. If two records match, data salvaging temporarily copies data from a passenger record to the driver record after comparing the two records. The copied data is data that is found in the passenger record but is missing or incomplete in the driver record. Data salvaging prevents blank matching or initials matching from matching records that you may not want to match.
Perform data salvage default	Specifies the default value that indicates whether to perform data salvage if the data record does not contain a field with this value. *Yes:* Performs the data salvage on the driver record after it matches a passenger record. *No:* Does not perform the data salvage on the driver record after it matches a passenger record.
Specify data salvage by field	Select to control data salvaging by means of a value in a field.
Perform data salvage field	Specifies the field that contains the indicator for performing the data salvage operation. Use this to override the default value.

Option	Description
Specify data salvage by source	Select to control data salvaging per source.
Source	Select a source from the drop-down menu. You must populate the Logical Input Source window with sources to have any appear in this drop-down list.
Perform data salvage	Select Yes or No to signify you want data salvaging performed on a source.

Related Information

Designer Guide: Match, Data Salvage [page 543]

3.5.4.12.2.3 Input Source options

Use the Input Sources options to define input sources for which you want to track statistics and to provide additional functionality throughout the Match transform. Before you define your input sources, you will need to map a field that contains the value that identifies the input source.

Option	Description
Value field	Choose an input field from the drop-down list that contains the value for your sources.
Source name	Enter a name for your input source.
Source value	Enter the value from the input data that identifies records belonging to this source. Sources are created by matching this case-sensitive value to that contained in the value field specified.
Source type	Choose a source type from the drop-down list. *Normal:* A Normal source contains good or eligible records. *Suppress:* A suppression source contains records that would often disqualify a record from use. For example, if you're using the software to refine a mailing list, a Suppress source removes records from the mailing. Examples: ● DMA Mail Preference File ● American Correctional Association prisons/jails lists ● No pandering or non-responder lists ● Credit card or bad-check suppression lists *Special:* A Special source is treated like a Normal source, with one exception. A Special source is not counted in when determining whether a match group is single-source or or multi-source. A Special source can contribute records, but it's not counted toward multi-source status.

Option	Description
	For example, some companies use a source of seed names. These are names of people who report when they receive advertising mail, so that the mailer can measure mail delivery. Appearance on the seed source should not be counted toward multi-source status.
Default source name	Specifies the name of a source to assign records to that do not belong to a predefined source. This name must match the name of a predefined source, so you must define input sources first to see any items in this list. The default source name will also be used if the Source value is blank or if the maximum number of sources is reached (maximum of 10,000).
Auto generate sources	Select to create sources for each unique entry in the Value field. This can save you time because you won't have to manually define your input sources. The name of an automatically generated source will be the same as the value in the Value field. As each record is processed, Match will first check to see if the record belongs to a predefined source. If it does, Match will assign that record to that source. If the record does not belong to a predefined source, then Match will check to see if the record belongs to an auto-defined source. If the record belongs to an auto-defined source, Match will use the auto-defined source. If the input source is not defined, Match will add the definition to the list of defined sources. If the maximum number of source definitions has been reached, then instead of adding a new source definition, Match will use the default source. ℹ Note When using auto generate sources, the Source ID is case sensitive but, the Type field is not case sensitive. Auto generate sources will accept Type field values of N, n, P, p, S, or s.
Default type	See the description of the Source type option in this table for information about source types. This type will be used for any source that does not already have a type defined in the Type field.
Type field	Choose an input field from the drop-down list that contains the input source type. The type field must have a value of: *N:* Normal *P:* Suppress (or Purge) *S:* Special If the field is not defined or if the field value is not N, n, P, p, S, or s, the default type is used.

Option	Description
	ℹ Note This option is not case sensitive.

3.5.4.12.2.4 Source Group options

Adding a Source Group operation can provide you with additional statistics in certain Match reports.

Option	Description
Source groups	Create your source groups here. To get started, double-click the checkbox in the first row, and type in a name for your source group.
Sources	This list contains all of the sources defined in the Input Sources operation.
Undefined action	Specifies the action to take if an input source does not appear in a source group. *Ignore:* The input source does not belong to any source group. *Default:* The input source belongs to the default source group specified in the *Default source group* option. *Auto:* ● If the *Source group field* option is not defined, then the input source will belong to a source group of the same name as the input source. The source group is created if necessary. ● If the *Source group field* option is defined, then the input source belongs to the source group named in the *Source group field* option. The source group is created if necessary. ● If the source group field's content is blank, then that input source will not belong to a source group (equivalent of Ignore).
Default source group	Select a default source group name from the drop-down list. You can choose from your defined source groups. This option is required if you chose Default as the value for the *Undefined action* option.
Source group field	Specifies the field that contains the value for your source groups.

3.5.4.12.3 Group forming

Group forming allows you to group and prioritize records for better match accuracy and efficiency.

Break groups

Break groups allow you to group records based on common field values (for example, postal code). Only records that share the same break group value will be compared with one another.

Use break groups to lower the number of comparisons needed and to increase the speed of the matching process.

Candidate selection

The process of candidate selection appends records from a relational database to an existing break group for processing.

For real-time jobs, candidate selection pulls a candidate set of records based on a single record or many records.

> i Note
>
> Candidate selection works with relational databases only; it does not work with flat files.

Group prioritization

Use group prioritization to ensure that your most complete and accurate records drive the comparison process.

Related Information

Break group options [page 1270]
Candidate selection options [page 1272]
Group prioritization options: Priority Order tab [page 1140]

3.5.4.12.3.1 Break group options

Use the break group options to group records based on common field values.

Options	Description
Split records into break groups	Select this option if you want to form break groups to reduce the total number of comparisons made. The most common case for deselecting this option is when have a real-time job and your data comes in as one break group. This scenario also often makes use of candidate selection (selecting a limited number of records from a relational database) for optimal real-time matching.

Options	Description
	⚠ Caution Deselect this option with caution within a batch data flow. The size of a break group may not exceed 2 GB. If you use this option in a batch data flow, also set the *Maximum allowable break group size (in records)* option so that the collection does not exceed the size limit. If it does exceed the limit, the data flow will abort. i Note Break group size is calculated by multiplying the record length by the number of records in the break group.
Field	Choose a mapped input field name from the drop-down menu that you want to include in the break key. Click the Add Row button to add another field. If you require a more complex break key, you could define that field using an upstream Query transform and select the field here.
Start Position	Enter the start position of the field. Valid values for a field of n are 1 to n and -1 to -n. Negative start values signify that the start position is counted from the right. For example, a field with a length of 7 contains JOHNSON. A start position of 2 would mean start with "O." A start position of -4 means start with the "N" (This would also be the case if the field has a length of 20, because the negative start value starts from the actual length of the string, not of the field).
Length	Enter the number of characters in the field you want included in the break key.
Break key case sensitive	Specifies whether to treat the break key as case sensitive. *Yes:* Treat the break key as case sensitive. *No:* Do not treat the break key as case sensitive. For example, if you create a break key using the primary name (street), separate break groups would be formed with values of “Main” and “main” when you specify that the break key is case sensitive.
Replace NULL with empty string	Specifies whether to convert NULL values with an empty string in the break key. *Yes:* Convert NULL to an empty string. *No:* Do not convert to an empty string.
Right pad fields with blanks	Because the break key is used for sorting and aggregating, it is sensitive to the position in which data is placed. By right-padding the break key fields you can help ensure that break groups are formed properly.

Options	Description
	If the *Replace NULL with empty string* option is set to YES and this option is set to YES, then fields with NULL values will be replaced with all spaces (to the length of the field). *Yes:* Right-pad fields with blank spaces. *No:* Do not right-pad fields.
Input already sorted	Specifies that the input data has already been sorted, and you do not want it sorted again. For example, if you require a more complex break key, you could use a Query transform to create it, and use the ORDER BY operation to order your data. *Yes:* The transform will not re-sort the input data. *No:* The transform will sort the break keys at runtime before forming break groups.
Maximum allowable break group size (in records)	Specifies the maximum number of records allowed in a break group. An empty value or zero means that there is no limit on the break group size. With this option, you can control the amount of memory used during processing by specifying the number of records processed at one time. If more records make it into a single break group than specified, then the data flow throws an error and stops.

Related Information

Designer Guide: Match, Break keys and candidate selection [page 533]

3.5.4.12.3.2 Candidate selection options

The candidate selection option group includes the following options:

Option	Description
Datastore	Select a valid datastore. This list is populated with all valid SQL and persistent cache datastores. If you choose a persistent cache datastore, you will not be able to enter custom SQL.
Cache type	This option can be used to improve performance, with a trade-off of more memory consumption. *No_Cache:* Specifies that each query will be sent to the database.

Option	Description
	Pre_Load_Cache: Specifies that the entire secondary table is cached to a local disk or memory.
Auto-generate SQL	Select to have your SQL generated by the transform. This option allows you to query a simple single table. If you need to join tables or create a complex WHERE clause, you should select the *Create custom SQL* option.
Table	Enter a valid table name from the datastore.
Use break column from database	Select this option if your database already contains a column that corresponds to the break key field.
Break key field	Select the column from the secondary table that contains the break key field.
Create custom SQL	Select to create custom SQL.
Launch SQL Editor	Opens the SQL editor. This button is only enabled if you select the *Create custom SQL* option.
Use constant source value	Select to assign records to a physical source for generating appropriate statistics.
Physical source value	Type a value for your physical source. This value will be placed in the physical source field you select.
Physical source field	Select the mapped field that contains the physical source name.
Add DB columns to mapping table	If you are using the *Create custom SQL* option, clicking this button will add only the database columns that appear in the SELECT statement and in the order that they appear in the SELECT statement. If you are using the *Auto-generate SQL* option, clicking this button will add ALL database columns, in the order that they appear in the table schema. Note If you do not associate an input field to any of these columns in the column mapping table, they will be removed when you close the window.

Column mapping table

This table allows you to specify which mapped field in the data flow each database selected field is assigned to.

Column	Description
Break key	Specifies whether this field is used as part of your break key.
Field	Each cell contains a list of the mapped names from the input fields in the transform.

Column	Description
DB column	Each cell contains a list of the column names in your database table or the selected columns from a custom query. Match the data of a column in your database to the data of a mapped field.

Related Information

Designer Guide: Match, Break keys and candidate selection [page 533]

3.5.4.12.3.3 Group prioritization options: Priority Order tab

Group forming prioritization

Use the Group prioritization operation to order records within each break group, which controls which records are used as the drivers during the comparison process.

Post-match prioritization

Add a Group prioritization operation before a Group Statistics operation to order records within a match group to control which record is flagged as the master record of each group of matching records. Add a Group prioritization operation before a Best Record operation to order records within a match group to control the destination of data that is being propagated from other records to form a best record.

Option	Description
Prioritization name	Specifies the name for this Group prioritization operation. If you have multiple operations in this Match transform, be sure to make this name unique.

Priority fields

Use the Priority fields table to order your break groups based on the content of a field (for example, dollar amount or date). Use the buttons to add, remove, and order rows. Place the primary sort field at the top of the list. The rest of the fields, in the order that they are positioned, determine the sub-sort that occurs.

Option	Description
Input field	Choose a field to sort your records on.
Field order	Specifies in which order records should be sorted.

3.5.4.12.3.4 Group prioritization options: Record Completeness tab

Option	Description
Prioritization name	Specifies the name for this Group prioritization operation. If you have multiple operations in this Match transform, be sure to make this name unique.
Order records based on completeness of data	Select this option to apply priority and blank penalty points to records to help control the order of your records.
Define only field penalties	Select this option when penalties need to be assessed based on blank fields.
Define priority and penalty fields	Select this option when you have specific fields that contain the actual integer values for priority and blank penalty.
Record priority field	Choose the field that contains priority values. This field must contain an integer.
Apply blank penalty field	Choose the field that contains the indicator (Y or N) for applying blank penalty points to a record.
Define priority and penalty based on input source	Select to have your record priority and blank penalty indicator (Y or N) determined by membership in a given source.
Source Name	Choose an input source from the drop-down list in the Source Name column. The sources listed here are defined in the Input Source operation.
Priority	Type a priority value (an integer) in the Priority column. Remember that the lower the priority score, the higher the priority.
Apply Blank Penalty	Choose Yes or No to determine whether a blank penalty is applied to a record based on membership to this source.
Default record priority	Specifies the default value for the record priority if the record does not contain a field with this value, this field is blank for a record, or if a record does not belong to any of the sources specified. Remember that the lower the priority score, the higher the priority.
Default apply blank penalty	Specifies the default indicator as to whether to add blank penalty points to records with blank fields. This indicator is used if a record does not have a field that carries this indicator, if that field is blank or has invalid data, or ifa record does not belong to any of the sources specified. *Yes:* Each record's blank penalty is added to the record's record priority to generate an adjusted record priority score. The lower the score, the higher the priority. *No:* No penalty is applied when the fields are blank.
Input field	Displays the input fields available to assign a blank penalty score to.
Blank penalty	Assign a penalty value (an integer) to apply when the specified field is blank in a record.

3.5.4.12.4 Matching

3.5.4.12.4.1 Match level options

These options affect processing at the match level only.

Table 164: Person Options

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>*Match level name*</td><td>Enter a name for this match level. Be sure that this name is unique within the Match transform.</td></tr>
<tr><td>*Weighted match score*</td><td>Specifies the weighted match score for this level.
When your matching method includes weighted scoring, records are considered matches when the total contribution score is greater than or equal to this value.</td></tr>
<tr><td>*Number of names that must match*</td><td>Specifies the number of names that must match. This option requires that you have criteria of Person1_Given_Name1, Person1_Family_Name1, and so on.
One: Specifies that records are a match when at least one of the names meet the criteria.
All: Specifies that records are a match only when all of the names meet the criteria.</td></tr>
<tr><td>*Match on hyphenated family name*</td><td>Specifies whether a single family (last) name in one record matches a hyphenated family name in another record. For example, this option considers whether the two records shown below are matches.
This comparison is performed only if one field has a hyphen and the other does not.
<table>
<tr><th>Given</th><th>Family</th></tr>
<tr><td>Laura</td><td>Smith</td></tr>
<tr><td>Laura</td><td>Albers-Smith</td></tr>
</table>
This option works on a criteria named Family_Name1, Family_Name2, or Family_Name3.
Yes: The family names match as long as the single family name in one record matches one of the hyphenated family names in another record.
No: The hyphenated family name is considered a single family name and the comparison results in a low similarity, usually not meeting your family name criteria, resulting in a no-match.</td></tr>
<tr><td>*Compare Given Name1 to Given Name2*</td><td>Specifies whether the given name1 (first name) of one record is compared to the given name2 (middle name of another record).
For example, the two records shown below could be considered duplicate records if this option is set to Yes.</td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><table>
<tr><th>Given name1</th><th>Given name2</th><th>Family name1</th></tr>
<tr><td>John</td><td></td><td>Smith</td></tr>
<tr><td>R</td><td>John</td><td>Smith</td></tr>
</table>
To use this option, you must have criteria named Person1_Given_Name1 and Person1_Given_Name2, Person2_Given_Name1 and Person2_Given_Name2, and/or Person3_Given_Name1 and Person3_Given_Name2.
<i>Yes</i>: The Given_Name1 field of one record is compared to the Given_Name2 field of another record.
<i>No</i>: The Given_Name1 field of one record is not compared to the Given_Name2 field of another record.</td></tr>
<tr><td><i>Ignore family name when female</i></td><td>Specifies whether an adjustment occurs for family names when the given name is a female. To use this option, you must have at least these three criteria: Given_Name1, Family_Name1, and Gender.
<i>Yes</i>: The Family_Name1 criteria is ignored when the given name gender is a female (Gender=5). For example, Laura Smith may match Laura Albers.
<i>No</i>: The gender is not used and the matching process is performed as usual.</td></tr>
</table>

Table 165: Address Options

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td><i>Match on Street and RR, or on Box</i></td><td>Specifies whether to match on PO Box only or on street, rural route, and PO Box. This option affects business and household records matching on address.
<i>Yes</i>: Records are considered a match if the Boxes match. If the Boxes do not match, then the address and rural route address must pass the match criteria settings.
<table>
<tr><th>firm</th><th>number</th><th>street</th><th>suffix</th><th>postal number</th><th>postcode</th></tr>
<tr><td>Acme Hardware</td><td>100</td><td>Elm</td><td>Ave</td><td>200</td><td>02961</td></tr>
<tr><td>Acme Hardware</td><td>123</td><td>Main</td><td>St</td><td>200</td><td>02961</td></tr>
</table>
<i>No</i>: All forms of the address (street, rural route, and PO Box) must match.</td></tr>
<tr><td><i>Address matches blank if Firms match</i></td><td>Specifies whether to match on firm data when other address data does not match. This only affects records when one has street information and the other has PO Box information. If both records have Street information that do not match, or if both have PO Box information that do not match, the records will not be found as duplicates.</td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><table>
<tr><th>firm</th><th>number</th><th>street</th><th>suffix</th><th>postal number</th><th>postcode</th></tr>
<tr><td>Acme Hardware</td><td>100</td><td>Elm</td><td>Ave</td><td></td><td>02961</td></tr>
<tr><td>Acme Hardware</td><td></td><td></td><td></td><td>300</td><td></td></tr>
</table>
<p><em>Yes</em>: If firm data matches and neither firm field is blank, blank matching is allowed for all address components.</p>
<p><em>No</em>: If firm data matches, but address data in one of the records is blank, the records will not be considered a match (unless blank matching is turned on for those address components).</p></td></tr>
<tr><td><em>Unique on resident if RR, but no Box</em></td><td><p>Specifies whether to match on Rural Route when an input record's Family Name field contains a resident-type name of Current Resident, Occupant, blank, or name not defined and a rural route address, with no box number.</p>
<p><em>Yes</em>: Places all records with this type of name data into the same match group.</p>
<p><em>No</em>: Does not place records with this type of name data into the same match group.</p></td></tr>
<tr><td><em>Ignore Firm if Name matches</em></td><td><p>This option works with odd abbreviations or spellings of firm names. This assumes that you are matching on two Family Names.</p>
<p><em>Yes</em>: Indicates matching names at the same address are matches, even if the firms don’t match. This lets you catch the following match, which might otherwise have been missed.</p>
<p>Rita Terranova Greenco 100 Bren Rd 55343</p>
<p>Rita Terranova Eco Technologies 100 Bren Rd 55343</p>
<p><em>No</em>: Both the firm criteria and the address criteria must meet the minimum similarity threshold in order to match.</p></td></tr>
</table>

Related Information

Designer Guide: Match, Weighted scoring method [page 550]

3.5.4.12.4.2 Match criteria table

Use the match criteria table to navigate to a particular criteria by double-clicking a row in the table.

Use the Add, Remove, and Move buttons to adjust the quantity and order of your match criteria.

3.5.4.12.4.2.1 Match criteria options: Criteria Fields tab

Available criteria

Choose a criteria that best reflects the data in the field you want to compare.

Category	Description
Geographic	*Address_Data1-5:* Use for address data that is not accounted for in other address-based criteria in the Geographic category. You can also use this criteria for fields that you know contain address data, but you're not sure which type it contains, or you can use it for international data that has not been parsed.
	Address_Post_Office_Box: Post Office box number.
	Address_Primary_Name: Street name data.
	Address_Primary_Number: Street number data.
	Address_Primary_Postfix: Address data that comes at the end of a street name, such as a directional.
	Address_Primary_Prefix: Address data that comes at the beginning of a street name, such as a directional.
	Address_Primary_Type: Data that tells what type of street it is (street, boulevard, lane, and so on).
	Address_Private_Mail_Box: A private mail box (PMB) number. These are mail boxes that are not run by a postal authority.
	Address_Rural_Route_Box: Rural-route box number (number only, without “Box” prefix).
	Address_Rural_Route_Number: Rural route number.
	Address_Secondary_Number: The number of a unit, building, floor, or room.
	Country: Country name.
	Locality: City, town, locality, or suburb.
	Latitude_Longitude: Latitude and longitude.
	Postcode1: Primary postal code.
	Postcode2: Secondary postal code.
	Region: Region data, such as state or province.
Firm	*Firm:* Firm name.

Category	Description
	Firm_Data1-3: Use for firm data that is not accounted for in other firm-based criteria. You can also use this criteria for fields that you know contain firm data, but you're not sure which type it contains. You can also use this for international data.
	Firm_Match_Std1-6: Firm match standards. The data in these fields is generated by the Data Cleanse transform or other pre-Match transforms.
	Firm_Location: A location within a company or organization.
	Firm_Location_Match_Std1-6 Match standards for a location within a company or organization.
Person	*Name_Data1-3:* Use for name data that is not accounted for in other name-based criteria. You can also use this criteria for fields that you know contain name data, but you're not sure which type.
	Person1-3_Given_Name1: The given name1 (first name) of the persons.
	Person1_Given_Name1_Match_Std1-6: Given_Name1 (first name) match standards for the first person.
	Person2_Given_Name1_Match_Std1-6: Given_Name1 (first name) match standards for the second person.
	Person3_Given_Name1_Match_Std1-6: Given _Name1 (first name) match standards for the third person.
	Person1-3_Gender: Gender.
	Person1-3_Family_Name1: Family (last) name.
	Person1-3_Family_Name1_Match_Std1-6: Family_Name1 match standards for the persons in your data record.
	*Person1-3_Family_Name2:*Family (last) name. Use this field when family name data is split into two fields, for example, for cultures where they store the paternal family name and the maternal family name in different fields.
	Person1-3_Family_Name2_Match_Std1-6: Family_Name2 match standards for the persons in your data record.
	Person1-3_Maturity_Postname: Maturity postname.. For example, Sr. or Jr. (one standard per person).
	Person1-3_Maturity_Postname_Match_Std1-6: Maturity postname match standards for the persons in your data record.
	Person1-3_Given_Name2: Given name2 (middle name).

Category	Description
	Person1_Given_Name2_Match_Std1-6: Given_Name2 (middle name) match standards for the first person.
	Person2_Given_Name2_Match_Std1-6: Given_Name2 (middle name) match standards for the second person.
	Person3_Given_Name2_Match_Std1-6: Given_Name2 (middle name) match standards for the third person.
	Person1-3_Honorary_Postname: Honorary postname for up to three persons indicating certification, academic degree, or affiliation. For example, CPA.
	Person1-3_Honorary_Postname_Match_Std1-6: Honorary postname match standards.
	Person1-3_Prename: Prename (for example, Mr. or Mrs.) for up to three persons.
	Person1-3_Prename_Match_Std1-6: Prename match standards.
	Person1-3_Title: Job or occupational title of each person. For example, Manager.
	Person1-3_Title_Match_Std1-6: Title match standards for each person.
	Social_Security_Number1-3: Social Security numbers for up to three people in a record.
Other	*Date1-3:* Date data. For example, birthdate data.
	Phone: Phone number.
Custom	Use custom fields to match data that does not qualify for any of the specifically named criteria. If you prepared fields for matching through a custom Universal Data Cleanse solution, the Data Cleanse dictionary names appear in the Custom category.

Criteria field mapping

Option	Description
Criteria field	The criteria field that contains the data you want to compare.
Input field mapped name	Choose the mapped name for your criteria field. If you choose any of the following input fields, the Match transform automatically adds the appropriate match standard fields. The fields displayed will vary depending on the value chosen for *Compare data using* on the Options tab. ● Firm ● Firm_Location

Option	Description
	- Person*_Given_Name1 - Person*_Given_Name2 - Person*_Honorary_Postname - Person*_Maturity_Postname - Person*_Prename - Person*_Title You can then choose to include or exclude any of these from mapping. Note If you enable multiple field matching, any appropriate match standard fields are removed. If you want to include them in the match process, add them in the *Additional fields to match* table in the *Multiple field matching* section.

3.5.4.12.4.2.2 Match criteria options: Options tab

Option	Description
Compare data using	Specifies how to handle fields where more than one word commonly exists. Only those options appropriate for the chosen value appear in the Comparison Rules table. *Field Similarity:* If you choose Field Similarity, the transform compares the entire field's data as a single string. This algorithm is more efficient and should be used in fields that typically have just one word. *Geo Proximity*: If you choose Geo proximity, the transform compares latitude and longitude information from different records for geographic proximity to determine if they are close enough to be considered duplicates. *Numeric Difference*: If you choose Numeric Difference, the transform compares numeric information from different records based on numerical difference to see if they are close enough to be considered duplicates. *Numeric Percent Difference*: If you choose Numeric percent difference, the transform compares numeric information from different records based on percentage of numerical difference to see if they are close enough to be considered duplicates. *Word Similarity:* If you choose Word Similarity, the transform first parses the data into words and then compares the words. This algorithm is less efficient than the Field algorithm, but will do a better job comparing data that typically has more than one word in it. Many criteria options require this option to be set to Word Similarity.

Pre-comparison options

Use these options to alter the data used for the comparison process. These options do not alter the data that is output from the Match transform.

Option	Description
Field compare length	Specifies the number of characters in the field to compare. Note This option will be disabled (grayed out) if the input field mapped to the main criteria (not a match standard) is not of type varchar.
Remove punctuation	Specifies whether to remove punctuation from your data to help provide more accurate matches. Be aware of the following: - This option is valid for Latin1 data only. - Match will not remove a dash from a Family_Name* field. *Yes:* Removes punctuation. *No:* Keeps punctuation in your data. Note This option works on the mapped input field used in this and other criteria. If you set this option to something different than how it is set in another criteria using this same field, it will override that setting. Before setting this option, be sure that there are no other criteria using this same field. ⚠ Caution Setting this option and the *Convert text to numbers* option to *Yes* may produce undesirable results. For example: Suppose you have `1.23` as data in a criteria field. Setting *Remove punctuation* to *Yes* would convert this number to `123`. This number would then match another value of `123`, or, in the case of converting text to numbers, match a value of "one hundred twenty-three".
Convert to uppercase	Specifies whether to convert all data to uppercase for matching purposes only. This option is valid for Latin1 data only. *Yes:* Converts the data to uppercase for the comparison process. *No:* Preserves the case of the data for the comparison process, treating A and a as different characters.

Option	Description
	Note This option works on the mapped input field used in this and other criteria. If you set this option to something different than how it is set in another criteria using this same field, it will override that setting. Before setting this option, be sure that there are no other criteria using this same field.
Convert diacritical characters	Specifies whether to include diacritical characters in the matching process. Be aware of the following: • This option is valid for all match engine options. • This option works only on upper-Latin1 characters (values between 128 and 255). If you are processing Japanese data, for example, you may have some Latin1 data mixed in. In these cases you will be able to convert diacritical characters. *Yes:* Converts diacritical characters to the closest English ASCII equivalent for matching purposes. For example, ä converts to a. *No:* Preserves diacritical characters in the matching process, treating ä and a as not identical characters. Note This option works on the mapped input field used in this and other criteria. If you set this option to something different than how it is set in another criteria using this same field, it will override that setting. Before setting this option, be sure that there are no other criteria using this same field.
Convert text to numbers	Specifies whether numbers represented as text (one, two, three) should be converted to numbers. If you choose Yes, they will be in cardinal (one = 1) or ordinal (first = 1st) format. *Yes:* Converts numbers represented as text to numbers. *No:* Leaves any numerical text intact. Note This option works on the mapped input field used in this and other criteria. If you set this option to something different than how it is set in another criteria using this same field, it will override that setting. Before setting this option, be sure that there are no other criteria using this same field. Caution Setting this option and the *Remove punctuation* option to *Yes* may produce undesirable results. For example:

Option	Description
	Suppose you have `1.23` as data in a criteria field. Setting *Remove punctuation* to *Yes* would convert this number to `123`. This number would then match another value of `123`, or, in the case of converting text to numbers, match a value of "one hundred twenty-three".
Locale	Specifies the locale setting for this criteria field. If nothing is specified, or you specify *DEFAULT*, the default system locale in the software will be used. Setting this option is recommended if you plan to use the *Convert text to numbers* option.

Comparison rules

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td><i>Approx substring adjustment score</i></td><td>Specifies what score to give to words that were not matched to the other words in the compared string. This option is commonly used to compare family names that have varying representations in which the <i>Substring adjustment score</i> option is too strict to compare, such as "Cruz Rodríguez" and "C. de Rodríguez".
Enter a value from 0 (default) to 100. Enter a value of 0 to disable the option.</td></tr>
<tr><td><i>Abbreviation adjustment score</i></td><td>This option controls matching whole words to abbreviations. For example, long firm names are often abbreviated by removing letters. International Health Providers might be abbreviated to Intl Health Providers.
For example:
<table>
<tr><th>Full word</th><th>Possible abbreviations</th></tr>
<tr><td>Business</td><td>Bus, Bsnss, Bss</td></tr>
<tr><td>Database</td><td>Dat, Db, Dse</td></tr>
</table>
As shown in the examples, abbreviation means that the first letter of the shorter word matches the first letter of the longer word, and all remaining letters of the shorter word appear in the longer word in the same order as in the shorter word. The value you enter is the score given to the letters that are in the longer word but not the shorter word.
● Enter a value of 0 (zero) to disable abbreviation checking.
● Enter a value greater than 0 to enable this option.
● Enter a value of 100 if you want abbreviations and longer words to be considered a perfect match.</td></tr>
</table>

Option	Description
	ℹ Note For this option to work, you must set the *Compare data using* option to *Word Similarity*.
Both fields blank operation	Specifies whether to use this criteria when both of the records' fields for this criteria are blank. *Eval:* The value entered in the Both fields blank score option is used as the similarity score for this criteria. *Ignore:* This criteria is ignored in the comparison process, and its contribution to the weighted score is proportionally distributed among the remaining criteria, therefore negating any impact the contribution score may have had.
Both fields blank score	Specifies the similarity score if both of the fields are blank and Both fields blank operation is set to *Eval*. Enter a value from 0 to 100.
Check for transposed letters	Specifies whether the match score should be adjusted for any transposed characters encountered. *Yes:* The transform deducts half as many points for transposed characters as it deducts for other non-matching characters. For example: Comparison: Smith—Simth Finding: Words differ by one transposition (penalty of 1 correction) Percentage alike: 90% *No:* The transform handles transposed characters the same way it handles any non-matching characters. For example: Comparison: Smith—Simth Finding: Words differ by one transposition (penalty of 1 correction) Percentage alike: 80%
Contribution to weighted score	Specifies the contribution value, when you use the weighted or the combination scoring method. If no single criteria decides a match or no-match, the contribution score is calculated by summing the products of each criteria's score by each criteria's weight. Type a value between 0 and 100.

Option	Description
Distance Unit	Specifies the type of distance unit used to calculate the distance between two Latitude, Longitude pairs. This option is only available when the Geo Proximity option is selected. Select one of the following: ● *Feet* ● *Kilometers* ● *Meters* ● *Miles*
Enable inter-script matching	Enable this option if you have the same data in different scripts (writing systems). For example one record has Latin1 and other has Katakana, or one has Latin and other has Cyrillic .
Ext abbreviation adjustment score	This option handles a variation of the *Abbreviation adjustment score* option. Enter a number that adjusts the similarity score for these types of abbreviations. For example: ● Enter a value of 0 (default) to disable abbreviation checking. ● Enter a value greater than 0 to enable abbreviation checking. Remember the following when using this option: ● The first letter of the short word must match the first letter of the first word in the multiple-word string, and the remaining letters of the short word must be found in order in the multiple-word string. ● Letters that match are given a score of 100. The remaining letters are given the score that you specify. ● The two scores are proportionally combined to render the overall score. Note For this option to work, you must set the *Compare data using* option to *Word Similarity*.
Initials adjustment score	Specifies whether you want initials or acronyms to match whole words. For example, the firm name International Health Providers could match IHP. Enter a value from 0 to 100. Enter a value of 0 (default) to disable initial checking. Remember the following when using this option: ● The initial must match the first letter of the word. ● The letters that match are given a score of 100. The remaining letters are given the score that you specify (from 1-100). ● The two scores are proportionally combined to render the overall score. If there are other words in the field that are not shortened, they are scored the usual way. For example, New York Police Department may be shortened to New York PD and still match.

Option	Description
	ℹ Note For this option to work for multiple-word abbreviations (such as International Health Providers = IHP) you must set the *Compare data using* option to Word Similarity. For this option to work for single-word abbreviations (such as Maria = M) you may set the *Compare data using* option to either Word Similarity or Field Similarity..
Match score	Specifies the minimum similarity score needed for the records to be considered a match based on this criteria. Type a value from 0 to 101. A value of 101 ensures that this criteria alone is not enough to consider two records a match and that you want to consider other criteria in the comparison process. For example, a value of 90 means that you consider this data to be important enough that if the data in two records is 90% similar or higher, the records are considered a match.
No match score	Specifies the maximum similarity score needed for the records to be considered a no-match based on this criteria. Type a value from -1 to 100. A value of -1 ensures that this criteria is not enough to consider two records a no-match and that you want to consider other criteria in the comparison process. For example, a value of 49 means that if the similarity between the data in two records is less than 50%, the records do not match.
Max Difference	Specifies the maximum difference allowed in a numeric range. Type a value from 0 - 2147483647.
Max Difference Score	Specifies what score to generate when the difference is the same as the Max Difference. Valid values for this required attribute range from 0 to 100. Any difference larger than the Max Difference will receive a score of 0. A difference equal to Max Difference will receive a score of Max Difference Score. Any difference less than Max Difference will receive a proportional score between Max Difference Score and 100.
Max Distance	Specifies the maximum distance allowed when calculating the distance between two Latitude, Longitude pairs. Type a value from 0 to 4294967295.0.
Max Distance Score	Specifies what score to generate when the distance is the same as Max Distance. Type a value from 0 to 100. Any distance larger than the Max Distance will receive a score of 0. A distance equal to Max Distance will receive a score of Max Distance Score.

Option	Description
	Any distance less than Max Distance will receive a proportional score between Max Distance Score and 100.
Max Percent Difference	Specifies the maximum difference allowed as a percent of the absolute value. Type a value from 0 to 100.
Max Percent Difference Score	Specifies what score to generate when the difference is the same as Max Percent Difference. Valid values for this required attribute range from 0 to 100. Any difference larger than the Max Percent Difference will receive a score of 0. A difference equal to Max Percent Difference will receive a score of Max Percent Difference Score. Any difference less than Max Percent Difference will receive a proportional score between Max Percent Difference Score and 100.
Numeric words match exactly	Specifies how to match data that contains both numbers and letters. *None:* Numeric words don't need to match exactly to be considered a match. *Any_Position:* Numeric words don't need to be in the same position in two different strings to be considered a match. *Any_Position_Consider_Punctuation:* This value behaves the same as the Any_Position value. However, the Match transform takes the position of a decimal separator (comma or period) within the numeric words into consideration. *Any_Position_Ignore_Punctuation:* Same as Any_Position, except that decimal separators (comma or period) are completely ignored. *Same_Position:* Numeric words must match exactly and be in the same position in the string to be considered a match. Note For this option to work, you must set the *Compare data using* option to *Word Similarity*.
One field blank operation	Specifies whether to use this criteria if one record's field is populated and the other record's field is blank. *Eval:* The value entered in the One field blank score is used as the similarity score for this criteria. *Ignore:* This criteria is ignored in the comparison process, and its contribution to the weighted score is proportionally distributed among the remaining criteria.
One field blank score	Specifies the similarity score to use if one of the fields is blank and the *One field blank operation* option is set to *Eval*. Type a value from 0 to 100.

Option	Description
Substring adjustment score	Allows matching longer strings of words to shorter strings. For example, long firm names are often shortened to the first few words of the name. A fictitious company such as Mayfield Painting and Sand Blasting might be shortened to Mayfield Painting. Remember the following rules about values to enter: - Enter a value from 0 to 100. Enter a value of 0 (default) to disable substring checking. - Enter a value of 100 if you want substrings and longer strings to be considered a match. Here is what happens after processing. - Letters that match are given a score of 100. The remaining letters are given the score you specify (from 1-100). - The two scores are proportionally combined to render the overall score. To qualify as a substring match, the shorter string must exactly match the first part of the longer string. Consider the following example: Matching substrings - Mayfield - Mayfield Painting - Mayfield Painting and Sand Substrings that do not match - Mayfield Sand Blasting - Painting and Sand Blasting Alternate spellings in any of the words also disqualify the substrings as a match. For example, “Murphy Painting and Sand Blasting” does not match. (This comparison would have a similarity score of 85% without this option set.) Note For this option to work, you must set the *Compare data using* option to *Word Similarity*.
Use in weighted score if greater than	Specifies the minimum similarity score needed to qualify this criteria to contribute to the Weighted Match Score. For example, if the value entered here is 59 for a given name criteria, and the given names between two records are less than 60% similar, then the given name criteria is ignored and the contribution value is proportionally distributed among the remaining criteria.
Zero weighted score if less or equal	Specifies the minimum similarity score needed for this criteria to qualify for contributing a value other than zero to the weighted match score.

Option	Description
	For example, if the value is 59 for the given name criteria, and the given names between two records are less than 60% similar, then the given name criteria contributes zero toward the weighted match score.

Reset Rules buttons

We provide you with default matching scores for every criteria (including options such as Match score, No match score, Contribution to weighted score, One field blank score, Both fields blank score, and so on). The Reset Rules buttons allow you to try different base levels of matching scores, as well as provide you a way to return to the default scores. These buttons primarily adjust the No match score and other options that are dependent on that score. The *Reset All to Default* button returns all options and scores to the original default values.

Loose matches mean that there will be more matches in a match group, but you may sacrifice in the quality of the matches.

Exact matches mean that the quality of the matches will be high, but they will be fewer in number.

Related Information

Designer Guide: Match, Numeric data matching [page 555]
Designer Guide: Match, Extended abbreviation matching [page 552]
Designer Guide: Match, Matching methods [page 548]
Designer Guide: Match, Unicode matching [page 580]

3.5.4.12.4.2.3 Match criteria options: Multiple Field Comparisons tab

Multiple field mapping

Option	Description
Compare multiple fields	Enable multiple field matching.
All selected fields in other records	Match each field against all fields selected in the table in each record.
The same field in other records	Match each field only against the same field in each record.

Additional fields to compare

Option	Description
Criteria field	Choose the input field that contains the data you want to compare.
Custom name	Enter a name for your custom field.
Input field mapped name	Choose the mapped input field name for the criteria field. The fields displayed will vary depending on the value chosen for *Compare data using* on the Options tab.

Pre-comparison options

Use these options to optimize your data for faster comparison. These options do not alter your source data; they allow Match to change your data internally.

Option	Description
Field compare length	Specifies the number of characters in the field to compare.
Remove punctuation	Specifies whether to remove punctuation from your data to help provide more accurate matches. Be aware of the following: ● This option is valid for Latin1 data only. ● Match will not remove a dash from a Family_Name* field. *Yes*: Removes punctuation. *No*: Keeps punctuation in your data. Note This option works on the mapped input field used in this and other criteria. If you set this option to something different than how it is set in another criteria using this same field, it will override that setting. Before setting this option, be sure that there are no other criteria using this same field. Caution Setting this option and the *Convert text to numbers* option to *Yes* may produce undesirable results. For example: Suppose you have `1.23` as data in a criteria field. Setting *Remove punctuation* to *Yes* would convert this number to `123`. This number would then match another value of `123`, or, in the case of converting text to numbers, match a value of "one hundred twenty-three".
Convert to uppercase	Specifies whether to convert all data to uppercase for matching purposes only. Be aware of the following:

Option	Description
	- This option is valid for English language, Latin1 data only. - This option is ignored for all other Match engine option values. *Yes*: Converts the output data to uppercase where appropriate. *No*: Leaves the output data intact. ℹ Note This option works on the mapped input field used in this and other criteria. If you set this option to something different than how it is set in another criteria using this same field, it will override that setting. Before setting this option, be sure that there are no other criteria using this same field.
Convert diacritical characters	Specifies whether to include diacritical characters in the matching process. Be aware of the following: - This option is valid for all match engine options. - This option works best when using the Latin1 engine. If you are processing Japanese data, for example, you may have some Latin1 data mixed in. In these cases you will be able to convert diacritical characters. *Yes*: Converts diacritical characters to the closest English ASCII equivalent for matching purposes. For example, ä converts to a. *No*: Preserves diacritical characters in the matching process, treating ä and a as not identical characters. ℹ Note This option works on the mapped input field used in this and other criteria. If you set this option to something different than how it is set in another criteria using this same field, it will override that setting. Before setting this option, be sure that there are no other criteria using this same field.
Convert text to numbers	Specifies whether numbers represented as text (one, two, three) should be converted to numbers. If you choose Yes, they will be in cardinal (one = 1) or ordinal (first = 1st) format. *Yes*: Converts numbers represented as text to numbers. *No*: Leaves any numerical text intact. ℹ Note This option works on the mapped input field used in this and other criteria. If you set this option to something different than how it is set in another criteria using this same field, it will override that setting. Before setting this option, be sure that there are no other criteria using this same field.

Option	Description
	⚠ Caution Setting this option and the *Remove punctuation* option to *Yes* may produce undesirable results. For example: Suppose you have `1.23` as data in a criteria field. Setting *Remove punctuation* to *Yes* would convert this number to `123`. This number would then match another value of `123`, or, in the case of converting text to numbers, match a value of "one hundred twenty-three".
Locale	Specifies the locale setting for this criteria field. If nothing is specified, or you specify *DEFAULT*, the default system locale in SAP Data Services will be used. Setting this option is recommended if you plan to use the *Convert text to numbers* option.

3.5.4.12.4.3 Compare table

The compare table options create a table that is used to determine which record pairs qualify to be compared or which sources should be compared. If you do not include a Compare table to the Match transform, a driver record is compared with all remaining passenger records in the break group.

> 🠚 Tip
>
> If you are using many physical or logical sources in your project, it may be easier to specify what not to compare, as opposed to what to compare. For example, say you have 10 sources: A through J. You want to compare all but A and B. Set the Default action option to Compare. Then set up a table row for both source A and source B, and set the Action options for those sources to No_Match.

Option	Description
Default action	Specifies the action assigned to each cell of the compare table initially before any Compare actions table rows are applied. *Compare:* Compares all of the logical sources specified in the compare table. You can then specify a pair to not be compared in the Compare actions table. *No_Match:* Does not compare all logical sources in the compare table. You can then specify a pair to be compared in the Compare actions table.
Default source	Specifies the source that a record belongs to if that record has no field to identify it. If there are no pre-defined sources, you can type a name.
Define compare actions using field values	Select to use values in a field as the criteria for comparison, rather then membership in an input source. To use this option, you will need to enter field values in the *Compare Actions* table.

Option	Description
	This option is automatically selected if you do not have any input sources defined.
Logical source field	Specifies the field that contains the logical source value (name).
Default logical source value	If you are not passing in a field from another location that contains the logical source value, or there is no value in the field, then you must specify the default value here. This option specifies which value defined in the Compare actions table is the default value.

Compare actions

Use this table to set the action of driver records and the records they are compared with (passenger records). These entries can override your Default action option value.

Option	Description
Driver source	Specifies the source contained in the driver record. If it is omitted, then all drivers records are assumed to be the first in a break group.
Passenger source	Specifies the value contained in the passenger record. If it is omitted, then all passenger records are assumed to be any record not the first in the break group.
Action	Specifies the action to take when a record from the Driver source is to be compared with a record from the Passenger source. *Compare:* Compare the two records *No_Match:* Do not compare the two records. This option overrides the Default action option. For example, if you set the Default action option to No_Match, you can specify that you want this pair to be compared by setting this option to Compare.

3.5.4.12.4.4 Post-match processing

Use the Post-match processing table to navigate to your operations by double-clicking the desired row in the table.

Best record

The purpose of best record post-processing is to salvage data from matching records—that is, members of match groups—and consolidate, or post, that data to a best record, or to all matching records.

Group statistics

Use group statistics to generate statistical information about your group of matching records. Find out:

- the number of records within the match group
- the sequential group order number
- the group rank, which flags one record within each group of matching records as the Master record and all other records in the group as Subordinate records
- whether the records in a match group belong to more than one source

Group statistics are essential for generating data for match reports.

Prioritization

Use the prioritization operation to order records for processing by other post-match operations.

Unique ID

You can use the Unique ID options to assign sequential identification numbers to each new record when adding records to a data warehouse. For example, the largest number assigned in a particular project can be carried over as the beginning identification number (plus 1) to be used in the assignment of new sequential IDs. This occurs when the software processes the next source against the data warehouse file.

Output record

Use the Output record options to flag certain types of records for potential processing downstream.

Related Information

Best record options: Best Record tab [page 1136]
Group statistics options [page 1139]
Group forming [page 1269]

Output flag selection options [page 1306]
Designer Guide: Match, Best record [page 562]
Designer Guide: Match, Unique ID [page 570]

3.5.4.12.4.4.1 Best record options: Best Record tab

Use the best record post-match processing operation to update your records with information from other records in a match group, among other things.

Option	Description
Best record name	Enter a name for your Best Record operation. Make sure that this name is unique within this Match transform.
Best record strategy	Choose the strategy to determine whether any action is taken on records in a match group. This is the criteria for further action. After you choose the strategy, priority, and field that you want to work with, the Match transform automatically generates the Python code for you (except for Custom). *Custom:* Choose this strategy to base your strategy entirely on custom Python code. This allows you to open the Python Expression editor and create custom Python code. *Date:* Choose Date to base your strategy on a date field. *Length:* Choose Length to base your strategy on the length of data in a field. *Non_Blank:* Choose Non_Blank to base your strategy on the completeness of data in a field. *Priority_Number:* Choose Priority_Number to base your strategy on a number. *Priority_String:* Choose Priority_String to base your strategy on string data in a field.
Strategy priority	These are the choices for priorities for each of the best record strategies, other than Non_Blank and Custom. Date • *Newest:* The newest date in the field will cause an action to take place. • *Oldest:* The oldest date in a field will cause an action to take place. Length • *Longest:* The longest string in a field will cause an action to take place. • *Shortest:* The shortest string in a field will cause an action to take place. Priority_Number • *Highest:* The highest number in a field will cause an action to take place. • *Lowest:* The lowest number in a field will cause an action to take place. Priority_String

Option	Description
	- *Ascending:* The string with the most ascending string order will cause an action to take place. - *Descending:* The string with the most descending string order will cause an action to take place.
Strategy field	Choose a field that contains data that you need to execute your strategy.
Posting destination	Specifies the destination record. *Master:* Post only to a master record. *Subs:* Post only to subordinate records. *Master to Subs:* Push information from the master record and post it to each subordinate record. *All:* Post to both the master and subordinate records.
Post only once per destination	*Yes:* Post only once per destination record. After data is posted to the destination record, the operation stops. *No:* Post more than once per destination. After data is posted to the destination record, the operation continues and the destination record is populated again with the next value. This option should be used when accumulating values such as total sales. Set this option to Yes when you are using the NON_BLANK strategy. Set this option to No when you are using the Longest, Shortest, Newest, Oldest, Ascending, or Descending priorities. Note This option is ignored when using the *Master to Subs* posting destination. With this posting destination, information can be posted only once.
View/Edit Python	The View/Edit Python button opens the Python Expression editor. If you chose the Custom strategy, you can create your custom Python code. If you chose any other strategy, Python viewed in the editor is read-only.

Best record action fields

Use the best record action fields table to define the actions taken on the fields based on your strategy.

Option/Option group	Description
Source field	Specifies the name of the source field in the input record.
Destination field	Specifies the name of the destination field in the output record, or the destination of this best record action.

Option/Option group	Description
	You can have the action post to the same input field, or you can post to a different field.
Custom	*Yes:* Specifies that you want to create custom Python code to perform an action on the destination field. *No:* Specifies that you want to use the same source and destination fields. When this option is set to No, the contents of the source field are copied to the destination field.
Editor	If you chose Yes in the Custom column, a button appears here to allow you open the Python Expression editor and configure your Python code. You can open the Python Expression editor only if Custom is set to Yes.

Related Information

Designer Guide: Match, Best record [page 562]

3.5.4.12.4.4.2 Best record options: Destination Protection tab

Protect data from being changed by enabling and defining destination protection.

Option	Description
Best record name	Enter a name for your Best Record operation. Make sure that this name is unique within this Match transform.
Enable destination protection	Select to protect records from best record operations that may modify the contents.
Default destination protection	Select the default destination protection setting. This is useful because the default setting will account for records that are protected (or not protected) through the use of sources or fields.
Specify destination protection by field	Select to enable destination protection through a value in a field.
Destination protection field	Choose the field that holds the destination protection value. The field must contain a value of Y or N. Any other value (including blank) will cause the default destination protection specification to occur, if you specified a default destination protection.
Specify destination protection by source	Select this option to control destination protection through membership in a particular source. Fill in the table with source names and whether they are protected.

Option	Description
Source Name	Choose the name of the source from the drop-down list. The list here is populated with defined sources and source groups from the *Input Sources Editor* window of the Match Editor.
Destination protected	Select a value to assign to the source. Select Yes to enable destination for that source. Select No, if you do not wish to protect records from that source.

3.5.4.12.4.4.3 Group statistics options

The Group Statistics option group includes the following options:

Option	Description
Group statistics name	Choose a name for this group statistics operation. If you are including more than one group statistics operation in this Match transform, make sure that the name is unique.
Generate only basic statistics	Select if you want to generate match group statistics. These will not include any statistics about input sources.
Generate source statistics from input sources	Select to generate statistic counts about your input sources. You must have input sources defined in the Match editor for this option to be active. If you do not check this, the Match transform will still generate statistics about your match groups.
Generate source statistics from source values	Select to generate source statistics based on source values in a field. If you have a source value field, using this option, you can choose to count all sources or specific ones based on a particular value.
Logical source field	Specifies the field that holds the value for your logical sources.
Default logical source value	Specifies a value to use if the field in the Logical source field option is blank. For example, if a record has a blank value in the field, this default value is used.
Count all sources	Select to count all sources, no matter what the value in the Logical source field is.
Choose sources to count	Select to specify particular sources to count, based on values in the Logical source field.
Default count flag value	Specifies the value to use when the Predefined count flag field is invalid (for example, if the field has data other than Y or N) or it is empty. *Yes:* Counts all of the records in the source. *No:* Does not count any of the records in the source.
Auto generate sources	Select to generate sources based on the value in a field.

Option	Description
Predefined count flag field	Specifies the field name that contains the indicator value (Y or N) to determine whether a source is counted.

Manually define logical source count flags

Be sure to fill in both columns for this to work.

Option	Description
Source value	Specifies the value in the field to find. This value is case sensitive.
Count	Specifies whether you want to use the value you entered in the Logical source value option in the count. *Yes:* Includes the logical source value in the count. *No:* Does not include the value in the logical source value option in the count.

3.5.4.12.4.4.4 Group prioritization options: Priority Order tab

Group forming prioritization

Use the Group prioritization operation to order records within each break group, which controls which records are used as the drivers during the comparison process.

Post-match prioritization

Add a Group prioritization operation before a Group Statistics operation to order records within a match group to control which record is flagged as the master record of each group of matching records. Add a Group prioritization operation before a Best Record operation to order records within a match group to control the destination of data that is being propagated from other records to form a best record.

Option	Description
Prioritization name	Specifies the name for this Group prioritization operation. If you have multiple operations in this Match transform, be sure to make this name unique.

Priority fields

Use the Priority fields table to order your break groups based on the content of a field (for example, dollar amount or date). Use the buttons to add, remove, and order rows. Place the primary sort field at the top of the list. The rest of the fields, in the order that they are positioned, determine the sub-sort that occurs.

Option	Description
Input field	Choose a field to sort your records on.
Field order	Specifies in which order records should be sorted.

3.5.4.12.4.4.5 Group prioritization options: Record Completeness tab

Option	Description
Prioritization name	Specifies the name for this Group prioritization operation. If you have multiple operations in this Match transform, be sure to make this name unique.
Order records based on completeness of data	Select this option to apply priority and blank penalty points to records to help control the order of your records.
Define only field penalties	Select this option when penalties need to be assessed based on blank fields.
Define priority and penalty fields	Select this option when you have specific fields that contain the actual integer values for priority and blank penalty.
Record priority field	Choose the field that contains priority values. This field must contain an integer.
Apply blank penalty field	Choose the field that contains the indicator (Y or N) for applying blank penalty points to a record.
Define priority and penalty based on input source	Select to have your record priority and blank penalty indicator (Y or N) determined by membership in a given source.
Source Name	Choose an input source from the drop-down list in the Source Name column. The sources listed here are defined in the Input Source operation.
Priority	Type a priority value (an integer) in the Priority column. Remember that the lower the priority score, the higher the priority.
Apply Blank Penalty	Choose Yes or No to determine whether a blank penalty is applied to a record based on membership to this source.
Default record priority	Specifies the default value for the record priority if the record does not contain a field with this value, this field is blank for a record, or if a record does not belong to any of the sources specified. Remember that the lower the priority score, the higher the priority.

Option	Description
Default apply blank penalty	Specifies the default indicator as to whether to add blank penalty points to records with blank fields. This indicator is used if a record does not have a field that carries this indicator, if that field is blank or has invalid data, or ifa record does not belong to any of the sources specified. *Yes:* Each record's blank penalty is added to the record's record priority to generate an adjusted record priority score. The lower the score, the higher the priority. *No:* No penalty is applied when the fields are blank.
Input field	Displays the input fields available to assign a blank penalty score to.
Blank penalty	Assign a penalty value (an integer) to apply when the specified field is blank in a record.

3.5.4.12.4.4.6 Unique ID options: Unique ID tab

Use the Unique ID options to assign sequential identification numbers to each new record when adding records to a data warehouse. For example, the largest number assigned in a particular project can be carried over as the beginning identification number (plus 1) to be used in the assignment of new sequential IDs. This occurs when the software processes the next source against the data warehouse file.

Note

Also see the Unique ID section for information about working with unique ID in a multi-server environment. Depending on the processing operation and starting value source you use, there could be limitations for using unique ID.

The Unique ID option group includes the following options:

Option	Description
Unique ID name	Enter a name for this Unique ID operation. If you are using other Unique ID operations in this Match transform, you may want to specify the name of the match transform and match level in this name to distinguish it from others.
Processing operation	Specifies the type of processing operation you want the application to perform. Valid values include: *Assign:* Assigns a new ID to unique records that need one, or assigns a new ID to all members of a group that don't have an ID. In addition, the assign operation copies an existing ID when a member of a match group already has an ID. For assign operations to work, all match group members must appear consecutively in one collection and must be in priority order (high to low).

Option	Description
	AssignCombine: Performs both an assign operation and a combine operation. All match group members must appear consecutively in one collection and must be in priority order (high to low). *Combine:* Combines the IDs of a match group when more than one ID is represented. All match group members must appear consecutively in one collection and must be in priority order (high to low). *Delete:* Removes unique IDs from records that have one, unless they are protected. *Split:* Splits the IDs of an ID group when more than one match group is represented. All ID group members must appear consecutively in one collection and must be in priority order (high to low).
Recycle unique IDs	Specifies whether unique IDs that were freed up during the delete operation should be used again in different records. You may want to recycle unique IDs if you have a limited amount available. Valid values include: *Yes:* Recycle freed-up unique IDs. *No:* Do not recycle freed-up unique IDs.
ID field	A field that holds a previously assigned unique ID. If this field is omitted, then it is assumed that no records have a unique ID.
Field	The starting unique ID is obtained from an input field. Be sure to map in a field from an upstream transform before you add this option.
Starting unique ID field	Choose the field that passes in the starting unique ID. If no Unique ID is received, the starting number will default to 1.
Constant value	The starting ID is specified as a positive whole number in the Starting value option.
Starting value	Indicates the starting unique ID value. Valid values range from 1 to UINT_MAX (unsigned integer max). The default value is 1.
Value from file	The starting Unique ID is read from the file specified in the File option.
File	Specifies the path and name of the file that manages unique IDs. A value is required here only when the Starting unique ID source option is set to File.
GUID	Uses the Globally Unique Identifier (GUID) as the unique ID. This is also known as the Universal Unique Identifier (UUID). The UUID variation used for unique ID is a time-based 36-character string with the format: `TimeLow-TimeMid-TimeHighAndVersion-ClockSeqAndReservedClockSeqLow-Node`. For more information about UUID, see the RFC document in the Related Topics section.

Option	Description
Save ending ID to file and reclaim recycled IDs	Specifies whether to save the last ID that was assigned to a file. Additionally, specifies whether to reclaim recycled IDs.
File	Specifies the file to write the last assigned ID to.
Allow multiple Match transforms to access unique ID file	Allows multiple Match transforms to access a shared unique ID file. When enabled, multiple data flows can access the same unique ID file, and single Match transforms can run in more than one process when the DOP setting is greater than 1. In addition, this allows multiple Match transforms within a single data flow to share a single unique ID file.
Number of IDs to get when accessing file	Specifies the number of IDs to retrieve from the unique ID file during each access. For example, with a setting of 100, the first process will access the file and retrieve IDs numbered 1-100. The next process with retrieve IDs numbered 101-200. If a process uses less than the number of retrieved IDs, the remaining IDs are written back to the unique ID file as recycled IDs. Note A setting greater than 1 improves performance when sharing a unique ID file between multiple processes by reducing the number of times the file must be accessed. However, integer numbers may not be assigned in sequential order.
Group number field	Specifies the field that holds a group number. The group number is used to assign the same unique ID to more than one record. If this field is omitted, then it is assumed that each record is unique and should have its own number.

Related Information

Designer Guide: Match, Assign unique IDs using a file [page 570]
Designer Guide: Match, Unique ID [page 570]

3.5.4.12.4.4.7 Unique ID options: Destination Protection tab

Use the Destination Protection tab to control whether a record's unique ID is protected based on the source that the record belongs to. This can help prevent IDs from being assigned to a suppression or rented source.

Option	Description
Unique ID name	Enter a name for this Unique ID operation. If you are using other Unique ID operations in this Match transform, you may want to spec-

Option	Description
	ify the name of the match transform and match level in this name to distinguish it from others.
Enable destination protection	Select if you want to protect a destination source from having its unique IDs overwritten with the IDs from matching records.
Default destination protection	Select the default destination protection setting. This is useful because the default setting will account for records that are protected (or not protected) through the use of sources or fields.
Specify destination protection by field	Select to enable destination protection through a value in a field.
Unique ID protected field	Choose an input field from the drop-down list that holds the value for specifying whether this ID is protected. The field must contain a value of Y or N. Any other value (including blank) will cause the default destination protection specification to occur, if you specified one.
Specify destination protection by source	Select this option to control destination protection through membership in a particular source. Fill in the table with source names and whether they are protected.
Source name	Choose the name of the source from the drop-down list. The list here is populated with defined sources and source groups from the *Input Sources Editor* window of the Match Editor.
Unique ID protected	*Yes:* This source is protected. *No:* This source is not protected.

3.5.4.12.4.4.8 Output flag selection options

Select types of records you want to flag on output based on each of the input sources. You may want to flag these records so that they will be available for writing to output. Downstream in a data flow, you can check the value of the Select_Record (Y/N) and decide whether you want to write it to output by using a Case transform, for example.

This is a repeatable operation.

Record type	Description
Output flag selection name	Enter a unique name for this operation that will allow you to identify it in a report and in your Select_Record output field. For example, suppose you have two two Output Flag Selection operations in this match level: DMA_Matches and Mail_List. Your output fields are then called: ● <match level name>_DMA_Matches_Select_Record ● <match level name>_Mail_List_Select_Record
Unique	Records that are not members of any match group. No matching records were found. These can be from sources with a normal- or special-type source.

Record type	Description
Single source masters	Highest ranking member of a match group whose members all came from the same source. Can be from normal- or special-type sources.
Single source subordinates	A record that came from a normal- or suppress-type source and is a subordinate member of a match group.
Multiple source masters	Highest ranking member of a match group whose members came from two or more sources. Can be from normal- or special-type sources.
Multiple source subordinates	A subordinate record of a match group that came from a normal- or suppress-type source whose members came from two or more sources.
Suppression matches	Subordinate member of a match group that includes a higher-priority record that came from a suppress -type source. Can be from normal- or special-type source.
Suppression uniques	Records that came from a suppress-type source, and for which no matching records were found.
Suppression masters	A record that came from a suppress-type source and is the highest ranking member of a match group.
Suppression subordinates	A record that came from a suppress-type source and is a subordinate member of a match group.

3.5.4.12.5 Input fields

The following are recognized input fields that you can use in the input mapping for the Match transform.

Name	Description
Option_Field_Algorithm_Geo_Proximity_**`<<logical_name>>`** _Max_Distance	The distance used in proximity matching. This setting is dynamic. If you change this setting, you do not have to terminate and reinitialize the transform in order for the new configuration to be recognized. **`<<logical-name>>`** is a name used to reference option groups. Some option groups can be repeated and the transform needs a way to uniquely identify each option group. In the Designer, this name is generated automatically by the Match editor. In order to understand which option is affected by this input field, look in the option tab of the Match transform and find the Field Algorithm Geo Proximity option group whose name is the same as appears in the input field.

Related Information

Designer Guide: Set up Geographic Proximity matching - Criteria options [page 560]

3.5.4.12.6 Output fields

The following Match fields are generated by the Match transform per match level. Use these fields when you map your output schema.

Field name	Default content type	Description
All_Match_Criteria	None	For internal use only
All_Match_Records	None	For internal use only
Break_Key	None	Specifies the break key generated for this record.
Conflict_Record	None	The sequence number of the record that has a conflict with this record. Conflict means that records that do not match directly are in the same match group. For example, records A and B match record C, but records A and B do not match directly, therefore the pair of records A and B are conflict records. For records that have no conflict, the value is blank. i Note Select this field for output only if your job was created by Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching (which is not currently available). Using this field under other conditions will result in output that is either blank or provides no value.
First_Match_Record	None	Specifies the sequence number of the record that this record first matched. Unique records will have a value of Null. This output field is useful when *Match Mode* is set to “Multi-Driver” to determine to which record a record first matched. i Note Select this field for output only if your job was created by Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching (which is not currently available). Using these fields under other conditions will result in blank output.

Field name	Default content type	Description
		Note Currently, the Match mode option is not available for edit in Data Services.
Group_Number	Group_Number	Specifies the records that belong to the same match group, which share the same group number. The group numbers start with the number 1. Unique records have a blank group number. If you are using association matching in your job, you need to map this on output, because the Associate transform uses it.
Match_Criterion	None	Specifies the name of the criteria or pattern that made the decision (if the Match_Type is *R*). Otherwise, the field is blank. Note Pattern matching is not currently available in Data Services
Match_Level	None	Specifies the name of the match level used.
Match_Pair_Count	None	Specifies how many records this record matched. A unique record will have a value of 0. A record that matched all the other records in a match group will have a value that is one less than the match group count. If Match Mode is set to “Single Driver”, then the driver record will have a match pair count equaling one less than the match group count, and the passengers will have a match pair count of 1. Note Select this field for output only if your job was created by Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching (which is not currently available). Using this field under other conditions will result in output that is either blank or provides no value. Note Currently, the Match mode option is not available for edit in Data Services.
Match_Score	None	The Match_Score field outputs the following values: ● The criteria or pattern similarity score when the Match_Type is *R*. ● The total weighted score when the Match_Type is *W*. ● Blank if the record is a driver record (Match_Type of *D*) or if the records are unique.

Field name	Default content type	Description
		ℹ Note Pattern matching is not currently available in Data Services.
Match_Status	None	The values for the Match_Status field that appear in your output are: *D:* This record is a driver in a match group. *P:* This record is a passenger in a match group. *U:* This is a unique record. *B:* Signifies that the *Match mode* option is set to "Multi-Driver" and the record is both a driver and passenger. ℹ Note Currently, the Match mode option is not available for edit in Data Services.
Match_Type	None	Indicates what type of match was made to bring the record into a match group. Possible values are: *<blank>:* The record did not match any other record. It is a unique record. *D:* The record was the driver record in the comparison process. *R:* The decision was made by a rule (criteria) or pattern. *W:* The record was identified as matching the driver because the total weighted score met the Weighted match score. See "Matching methods" in the *Data Services Designer Guide* for more information on weighted scores. ℹ Note Pattern matching is not currently available in Data Services.
Review_Record	None	The sequence number of the record that matches with this record but something about the pair of matching records is identified as questionable. For records that do not need review, the value is blank. ℹ Note Select this field for output only if your job was created by Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching (which is not currently available). Using this field under other conditions will result in output that is either blank or provides no value.

Field name	Default content type	Description
Sequence_Number	None	Each record input to the Match transform is assigned a sequence number. This number will be stored internally by a 64-bit integer. The first value assigned after the Match transform is initialized is 1, the second value is 2, and so on. The sequence number will allow Match to connect two records by having them reference each other's sequence number. The field data type is Decimal. Note Select this field for output only if your job was created by Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching (which is not currently available). Using this field under other conditions will result in output that is either blank or provides no value.

Input Source operation output fields

These fields are available only when you use an Input Source operation in your Match transform.

Field name	Description
Source_Group_Name	Specifies the name of the Source Group that the current record belongs to. If a record does not belong to any Source Group, then an empty string is output.
Source_Name	Specifies the name of the input source that the current record belongs to.
Source_Type	Specifies the type of source that the current record belongs to. *N:* The record comes from a normal source. *P:* The record comes from a suppress source. *S:* The record comes from a special source.

If you also add a Group Statistics post-match operation, and select the *Generate source statistics from input sources* option, the following output fields are available (these are in addition to the fields generated by the Group statistics operation).

Field name	Description
Group_Count	Provides the total number of records in the match group. Unique records have a value of 1.
Group_Rank	Specifies whether the record is a master (M) or a subordinate (S). Unique records have an empty value.
Group_Source_Appearance	Specifies the order the input source appears in this match group. The first input source appearing in the match group receives a value of 1, the second Input

Field name	Description
	Source appearing will get 2, and so forth. Records that come from the same input source will receive the same Group_Source_Appearance value. Unique records have a value of 0.
Group_Source_Group_Order	Specifies the order of the records within the match group that have the same Group_Source_Group_Appearance value. The first occurrence receives a value of 1, the second occurrence receives a value of 2, and so on. Unique records receive a value of 0. Records in a match group not assigned to a source group will also get a value of 0.
Group_Source_Order	Specifies the order of the records within the match group that have the same Group_Source_Appearance value. The first occurrence receives a value of 1, the second occurrence receives a value of 2, and so on. Unique records have a value of 0.
Group_Source_Type	Specifies the type of source in the match group. This field will contain one of the following values: *M:* The records come from more than one input source (excluding records from Special sources). *P:* The records come from a Suppress source. (If the master record comes from a suppression source, then all records in the match group have a P. If the master record comes from a normal or special source, then the suppression record and all records after it have a P, but the records before the suppression record have an M or S.) *S:* The records come from a single input source . *<empty>:* The record is unique.
Multi_Source_Count	Specifies the number of sources represented in the match group (excluding the Special sources and Suppress sources and Normal sources that follow a Suppress source in the match group). The values of this field could range from 0 to the number of records in the match group. Unique records receive a value of 1, if from a Normal list, and 0, if from a Special source or a Suppress source.
Source_Count	Specifies the number of sources represented in the match group (regardless of the source types). The values of this output field could range from 1 to the number of records in the match group. Unique records will have a value of 1.
Source_Group_Count	Specifies the number of Source Groups represented in the match group. Records in the match group that do not belong to a Source Group are not counted. The value of this output field range from 0 to the maximum number of input sources (10,000). Unique records will have a value of 0 or 1.

If you also add a Group Statistics post-match operation, and select the *Generate source statistics from source values* option, the following output fields are available (these are in addition to the fields generated by the Group statistics operation).

Field name	Description
Group_Count	Provides the total number of records in the match group.

Field name	Description
	Unique records have a value of 1.
Group_Order	The master record receives a value of 1. Subordinate records receive a value of 2 through the number of records in the match group. You may control the order by including a Group Prioritization in the Post Match Operations. Unique records have a value of 0.
Group_Rank	Specifies whether the record is a master (M) or a subordinate (S). Unique records have an empty value.
Group_Type	Specifies whether a record contributed to the source count, and if so, whether there were other sources represented in the match group. *M:* Multiple sources. Records from multiple sources are represented in the match group (records from Special sources are not counted toward a multiple-source match group). *S:* Single source. All records in the match group come from a single source (records from Special sources are not counted toward a multiple-source match group). *P:* At least one record from a Suppression source is included in the match group. (If the master record comes from a suppression source, then all records in the match group have a P. If the master record comes from a normal or special source, then the suppression record and all records after it have a P, but the records before the suppression record have an M or S.)
Source_Count	Specifies the number of sources represented in the match group (regardless of the source types). The values of this output field could range from 1 to the number of records in the match group. Unique records will have a value of 1.
Source_ID	Specifies the logical source value. In most cases, this is the input source value. In other cases it is the default logical source value.
Source_ID_Count	Specifies the number of source IDs represented in the match group.
Source_Type_ID	Specifies the type of logical source. *N:* Normal source *P:* Suppress source *S:* Special source

Source group operation output fields

These fields are available only if you use a Source Group operation in your Match transform.

Field name	Description
Group_Source_Group_Appearance	Specifies the order the source group appears in this match group. The first source group appearing in the match group receives a value of 1, the second

Field name	Description
	source group appearing receives a value of 2, and so on. Records that come from the same source group will receive the same Group_Source_Group_Appearance value. Unique records receive a value of 0. Records in a match group not assigned to a source group will also get a value of 0.
Group_Source_Group_Order	Specifies the order of the records within the match group that have the same Group_Source_Group_Appearance value. The first occurrence receives a value of 1, the second occurrence receives a value of 2, and so on. Unique records receive a value of 0. Records in a match group not assigned to a source group will also get a value of 0.
Source_Group_Count	Specifies the number of source groups represented in the match group. Records in the match group that do not belong to a source group are not counted. The values of this output field could range from 0 to the number of records in the match group. Unique records receive a value of 0 or 1.
Source_Group_Name	Specifies the name of the source group that the current record belongs to. If a record does not belong to any source group, then an empty string is output.

Group statistics operation output fields

These fields are available only if you use a Group Statistics operation in your Match transform.

Field name	Description
Conflict_Group	Contains C if the record is part of a match group that includes a conflict. Conflict means that records that do not match directly are in the same match group. For example, records A and B match record C, but records A and B do not match directly, therefore the pair of records A and B are conflict records. For records that do not belong to a match group with any conflicts, the value is blank. Note Select this field for output only if your job was created by Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching (which is not currently available). Using this field under other conditions will result in output that is either blank or provides no value.
Group_Count	Provides the total number of records in the match group. Unique records have a value of 1.
Group_Order	The master record receives a value of 1. Subordinate records receive a value of 2 through the number of records in the match group. You may control the order by including a Group Prioritization in the Post Match Operations. Unique records have a value of 0.

Field name	Description
Group_Rank	Specifies whether the record is a master (M) or a subordinate (S). Unique records have an empty value.
Group_Type	Specifies whether a record contributed to the source count, and if so, whether there were other sources represented in the match group. *M:* Multiple sources. Records from multiple sources are represented in the match group (records from Special sources are not counted toward a multiple-source match group). *S:* Single source. All records in the match group come from a single source (records from Special sources are not counted toward a multiple-source match group). *P:* At least one record from a Suppression source is included in the match group. (If the master record comes from a suppression source, then all records in the match group have a P. If the master record comes from a normal or special source, then the suppression record and all records after it have a P, but the records before the suppression record have an M or S.)
Review_Group	Contains R if the record is part of a match group for which review is recommended. Review is recommended because the pair of matching records is identified as questionable. For records that do not belong to a match group with any review records, the value is blank. Note Select this field for output only if your job was created by Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching (which is not currently available). Using this field under other conditions will result in output that is either blank or provides no value.
Source_Count	Shows the number of logical sources in this match group. Unique records have a blank value
Source_ID	Specifies the logical source value. In most cases, this is the input source value. In other cases it is the default logical source value.
Source_ID_Count	Specifies the number of source IDs represented in the match group.
Source_Type_ID	Specifies the type of logical source. *N:* Normal source *P:* Suppress source *S:* Special source

Unique ID operation output fields

These fields are available only if you use a Unique ID operation in your Match transform.

Field name	Description
ID_Status	Specifies the status of the Unique_ID output field. This field generates the following values: *<blank>:* No change. The Unique_ID output field is the same as the Unique_ID input field. *D:* Indicates that the Unique_ID output field has a blank unique ID and that the old unique ID was deleted. *N:* Indicates that the Unique_ID output field has a new unique ID. *O:* Indicates that the Unique_ID output field is assigned an old (existing) ID. This happens when a record is combined.
Record_Operation	Specifies the operation that should be performed on the record, based on the input fields (except protected fields). This field generates the following values: *<blank>:* Does not call an operation. *A:* Assigns a unique ID to the record. *C:* Combines the record's unique ID. *D:* Deletes the record's unique ID. *S:* Splits the record's unique ID.
Unique_ID	Specifies the unique ID the Match transform generated for this record. If the record already has a valid unique ID, then the output field will output the same unique ID. If the Match transform does not assign a unique ID, the output field is blank.

Group prioritization output fields

The following output fields are available when you add a Group Prioritization operation to a Match transform

Field	Description
Priority_Value	The sum of all priority and blank penalty values defined in the Record Completeness tab of the Group Prioritization. If you do not order records using the Record Completeness tab, this field contains 0.

Output flag selection output fields

The following output fields are available when you add an Output flag election operation to a Match transform.

Field	Description
Select_Record	Specifies whether the current record should be selected or not, based on your selections in the Output Flag Selection Editor. Valid values of this output field are Y if the record should be selected and N if the record should not be selected.

3.5.4.13 USA Regulatory Address Cleanse

The USA Regulatory Address Cleanse transform identifies, parses, validates, and corrects U. S. address data according to the U.S. Coding Accuracy Support System (CASS). This transform can create the USPS Form 3553 and output many useful codes to your records. You can also run in a non-certification mode as well as produce suggestion lists.

> Note
>
> If an input record has characters not included in the Latin1 code page, the USA Regulatory Address Cleanse transform will not process that data. Instead, the software sends the mapped input record to the corresponding standardized output field (if applicable). No other output fields will be populated for that record. If your Unicode database has valid U.S. addresses from the Latin1 character set, the transform processes as normal.

If you perform both data cleansing and matching, the USA Regulatory Address Cleanse transform typically comes before the Data Cleanse transform and any of the Match transforms in the data flow. SAP recommends using a sample job or data flow that is set up according to best practices for a specific use case.

The USA Regulatory Address Cleanse transform has several sample transform configurations that can help you set up your data flow. The transforms include all of the required options except input fields.

Related Information

Address Cleanse reference [page 1379]

Transform configurations [page 1123]

3.5.4.13.1 Content objects

Transform configurations

A transform configuration is a transform with preconfigured input fields, output fields, and options that can be used in multiple data flows. These are useful if you repeatedly use a transform with specific options and input and output fields.

When Data Services is installed, read-only transform configurations are provided for the Data Quality transforms.

You can use transform configurations in your data flows or as an example of a typical transform. After you place an instance of the transform configuration in a data flow, you can override these preset defaults. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one.

Sample blueprints and other objects

We have created Data Quality blueprints and other content objects to help you set up Data Services jobs. We've identified a number of common scenarios that you are likely to perform with Data Services. For each scenario, we've included a blueprint that is already set up to solve the business problem in that scenario.

Related Information

Downloading blueprints and other content objects [page 1128]
Transform configurations [page 1123]

3.5.4.13.2 Option groups

The USA Regulatory Address Cleanse transform includes options that you complete to process address data from the United States. The option groups are listed in the Related Topics list below:

Related Information

Report And Analysis [page 1318]
Reference Files [page 1319]
Transform Performance [page 1322]
USPS license information options [page 1323]
NCOALink options [page 1326]
Assignment Options [page 1333]
Standardization options [page 1336]
Non Certified Options [page 1342]
CASS Report Options [page 1344]
Suggestion List group [page 1345]
Z4_Change_Options [page 1349]

3.5.4.13.3 Report and analysis

Use these options to generate USA Regulatory Address Cleanse report data.

Option	Description
Gather Statistics Per Data Source	Specifies whether to generate report data per Data_Source_ID field value.

Option	Description
	Yes: Generates report statistics (if the *Generate Report Data* option is set to Yes) per Data_Source_ID field value. *No*: Generates reports (if the *Generate Report Data* option is set to Yes) for the input database without generating statistics based on the Data_Source_ID field value. i Note If you select Yes, other setup requirements apply. Read about statistics based on logical groups in the *Designer Guide*.
Generate Report Data	Specifies whether to generate report data for this transform. *Yes:* Generates report data for this transform. *No:* Turns off report data generation. If you do not need to generate reports (during testing, for example), you should set this option to *No* to improve performance.

Related Information

Designer Guide: Data Quality, Multiple data source statistics reporting [page 679]

3.5.4.13.4 Reference files

Reference files are directories used by the USA Regulatory Address Cleanse transform to correct and standardize your data. It is best to use the applicable substitution variable for the Option Value. This substitution variable represents the path to the reference files, and if you change that location, you can change the substitution variable so that all of your jobs reflect the current location.

Option	Description
Address Directory 1	`zip4us.dir` This directory, also called the National Directory, is organized by ZIP Code. It lists street names, ranges of house numbers, and postal and other codes. Use the substitution variable `$$RefFilesAddressCleanse`.
Address Directory 2	`*.dir` This second address directory is optional, and can be used for a customized address directory. No directory is automatically provided for this option. Most users should leave the Address Directory 2 option blank.
Address Geo 1 Directory	`ageo1.dir`

Option	Description
Address Geo 2 Directory *Address Geo 3 Directory* *Address Geo 4 Directory* *Address Geo 5 Directory* *Address Geo 6 Directory* *Address Geo 7 Directory* *Address Geo 8 Directory* *Address Geo 9 Directory* *Address Geo 10 Directory*	The Address-level GeoCensus files, `ageo1.dir` through `ageo10.dir`, are required only if you use the Address-level GeoCensus option or if you use the Geocoder transform. Use the substitution variable `$$RefFilesAddressCleanse`.
Address SHS Directory	`zip4us.shs` This directory enhances normal primary name lookups and is required for processing. Use the substitution variable `$$RefFilesAddressCleanse`.
Centroid Geo Directory	`cgeo2.dir` This directory is required only if you use the centroid-level GeoCensus option or if you use the Geocoder transform. Use the substitution variable `$$RefFilesAddressCleanse`.
City Directory	`city##.dir` The City directory is a table of city names, states, and ZIP Codes. It is organized by state and city. Use the substitution variable `$$RefFilesAddressCleanse`.
DPV Path	Specify the path to the DPV (Delivery Point Validation) directory files. These directory files are required for CASS certification. Use the substitution variable `$$RefFilesAddressCleanse`.
DSF2 Augment Path	Specify the path to the directory that contains the DSF2 (Second Generation Delivery Sequence) files you received from the USPS. Use the substitution variable `$$RefFilesAddressCleanse`.
eLOT Directory	`elot.dir` The eLOT directory contains eLOT codes for the delivery point that represents the mail carrier's delivery route walk sequence. Include this directory only if the *Enable eLot* option in the Assignment Options group is set to *Yes*. Use the substitution variable `$$RefFilesAddressCleanse`.
EWS Directory	`ewsyymmdd.dir`

Option	Description
	The software lists *ew*.dir* in the Option Value column, so that the transform finds the most current directory. Use the substitution variable `$$RefFilesAddressCleanse`.
LACSLink Path	Specify the path to the LACSLink directory files. These directory files are required for CASS certification. Use the substitution variable `$$RefFilesAddressCleanse`.
NCOALink Path	The location for the NCOALink directory files. These directory files are required for NCOALink processing and certification. Use the substitution variable `$$RefFilesAddressCleanse`.
Postcode Directory	`zcf10.dir` This directory contains the same data as the City directory, but is organized by ZIP Code. Use the substitution variable `$$RefFilesAddressCleanse`.
Postcode Reverse Directory	`revzip4.dir` The Reverse ZIP+4 directory enables the software to assign more postal codes when the input data includes a unique ZIP Code and valid ZIP+4. Use the substitution variable `$$RefFilesAddressCleanse`.
RDI Path	The RDI directory indicates whether an address is residential. Use the substitution variable `$$RefFilesAddressCleanse`.
Reverse Soundex Address Directory	`zip4us.rev` This directory enhances primary name lookups. Use the substitution variable `$$RefFilesAddressCleanse`.
SuiteLink Path	The SuiteLink directories contain specially indexed address information such as secondary numbers and unit designators for locations identified as high-rise business default buildings. These directory files are required for CASS certification. Specify the path to the SuiteLink directory files. Use the substitution variable `$$RefFilesAddressCleanse`.
USPS Log Path	Specify the path to the directory for NCOALink, DPV, and LACSLink log files. The provided substitution parameter is `$$Certification-LogPath`. The software determines the file names during processing as the USPS requires. This directory must already exist and be writable.

Option	Description
	It is important to use the same path for all jobs. If you have multiple clients, use the same log file directory for all clients so that the log files are combined.
Z4 Change Directory	`z4change.dir` The Z4Change directory lists all the ZIP Codes and ZIP+4 Codes in the country. Use the substitution variable `$$RefFilesAddressCleanse`.

For information about downloading directories, see the latest directories update.

3.5.4.13.5 Transform performance

The Transform Performance option group for the USA Regulatory Address Cleanse transform contains options that could improve the performance of DPV, DSF2, LACSLink, NCOALink, RDI, and SuiteLink processing.

Option	Description
Cache DPV Directories	Specifies whether the DPV directories are cached into memory. If the directories are cached, the caching takes place only once and is shared among all DPV threads running in the data flow. *Yes*: Enables caching. *No*: Disables caching.
Cache DSF2 Augment Directories	Specifies whether the DSF2 Augment directories are cached into memory. *Yes*: Enables caching. *No*: Disables caching.
Cache LACSLink Directories	Specifies whether the LACSLink directories are cached into memory. *Yes*: Enables caching. *No*: Disables caching.
Cache RDI Directories	Specifies whether the RDI directories are cached into memory. *Yes*: Enables caching. *No*: Disables caching.
Cache SuiteLink Directories	Specifies whether the SuiteLink directories are cached into memory. *Yes*: Enables caching. *No*Disables caching.

Option	Description
Insufficient Cache Memory Action	Specifies the action to take if there is insufficient memory to cache the directories that you have set up for caching. *Continue*: Attempts to continue initialization without caching. *Error*: Issues an error and terminates the transform.
NCOALink Caching Mode	Specifies the method for caching NCOALink directories. *Auto*: Select to have Data Services use available memory for caching. *Manual*: Select if you have a limited amount of memory available and you want to allocate a set amount of memory for caching. Enter a value in the *NCOALink Memory Usage* option. *None*: Disables caching. This is the default setting. Consider this option for smaller jobs because the overhead of caching directories could take longer than the actual job execution duration.
NCOALink Memory Usage	If the *NCOALink Caching Mode* is set to Manual, enter a value here to allocate a set amount of memory for NCOALink directory caching. The transform uses this value as the maximum amount of memory that can be used for the caching of NCOALink directories. i Note If the *Degree of Parallelism* value is greater than one, the *NCOALink Memory Usage* value is the total to be allocated for all threads. The value is not per thread.

3.5.4.13.6 USPS license information options

This group of options is required for all users performing NCOALink, SuiteLink, LACSLink, DPV, and DSF2 processing. You must provide information about the company performing the processing (the licensee) and the company for whom they are performing the processing (the customer). If you are performing the processing for yourself, you are the licensee and the customer.

The following table describes the *USPS Licensee and Customer Information Options* for the USA Regulatory Address Cleanse transform.

Option	Description
Customer Company Name *Customer Company Address* *Customer Company Locality* *Customer Company Region*	The customer is the person or company for whom you are performing NCOALink processing. End users may leave these fields blank unless you have an alternate stop processing agreement and have entered the special keycode into License Manager. The customer information appears in the NCOALink Processing Summary report and log files. The provided substitution parameters for these fields are: `$$CompanyName`

Option	Description

Customer Company Postcode1
Customer Company Postcode2
	`$$CompanyAddress`
	`$$CompanyLocality`
	`$$CompanyRegion`
	`$$CompanyPostcode1`
	`$$CompanyPostcode2`

Customer Company Phone	This is an optional field. The provided substitution parameter for this field is `$$CompanyPhone`.
DSF2 Licensee ID	Enter your DSF2 identification number as the USPS assigned it to you. You can use the substitution variable `$$DSF2LicenseeID` in this option.
IMB Mailer ID	This is an optional field. Enter your unique Intelligent Mail barcode (IMB) mailer ID that you received from the USPS, if applicable. The provided substitution parameter for this field is `$$IMBMailerID`. The IMB Mailer ID is a unique 6-digit or 9-digit numeric code assigned to mailers by the USPS based on their annual mail volumes. This information is included in the NCOALink Processing Acknowledgement Form (PAF).
Licensee Name	This field is required for NCOALink. The name of the company, up to 30 characters, as mentioned in the license agreement with the USPS. The licensee performs the NCOALink processing. This information appears in the PAF log and NCOALink Processing Summary report.
List Owner NAICS Code	This is a required field. Enter the North American Industry Classification System (NAICS) code, which identifies the business in which the list owner engages. The provided substitution parameter for this option is `$$CompanyNAICSCode`. For more information, visit the NAICS Web site at *http://www.census.gov/epcd/www/naics.html* .
List ID	The Customer or List ID is required for NCOALink limited and full service providers. End users may leave it blank. A unique ID assigned by the licensee to identify the list owner (customer). If the licensee does not have a naming scheme in place for the customer or lists, the six digits could be made up of the following: ● First 3 digits: Customer name/identifier ● Last 3 digits: List name identifier
List Processing Frequency	This 2-digit number (from 1 to 52) indicates how many times per year the list is processed with NCOALink. If the list owner has other lists processed by the NCOALink licensee at different frequencies, enter 99.

Option	Description
List Received Date	Enter the date when the NCOALink licensee received the list. Use the `yyyy/mm/dd` format. If you are an end user, you may leave this blank.
List Return Date	Enter the date when the list will be returned to the customer. Use the `yyyy/mm/dd` format. If you are an NCOALink end user, you may leave this blank.
Provider Level	This option lists the provider levels for which you have a registered license keycode. It defaults to the substitution parameter `$$USPSProviderLevel`. Only provider levels supported in your registered keycodes display in the option list.

3.5.4.13.6.1 Required options for USPS License Information

If you are processing NCOALink, DSF2, DPV, SuiteLink, or LACSLink, the USPS License Information group contains options that must be completed.

Tip

These options have substitution variables that you can set up in *Tools* › *Substitution Parameter Configurations*.

Option	NCOALink Full Service Provider*	NCOALink Limited Service Provider[2]	NCOALink End User*	DSF2	DPV	LACSLink
Licensee Name	X	X	X	X		
List Owner NAICS Code	X	X	X	X		
List ID	X	X	X	X		
Customer Company Name	X	X	X	X	X	X
Customer Company Address	X	X	X	X	X	X
Customer Company Locality	X	X	X	X	X	X
Customer Company Region	X	X	X	X	X	X
Customer Company Postcode1	X	X	X	X	X	X
Customer Company Postcode2	X	X	X	X	X	X
Customer Company Phone						

[2] With NCOALink enabled

Option	NCOALink Full Service Provider*	NCOALink Limited Service Provider[2]	NCOALink End User*	DSF2	DPV	LACSLink
List Processing Frequency	X	X	X			
List Received Date	X	X		X		
List Return Date	X	X		X		
Provider Level	X	X	X			
IMB Mailer ID						
DSF2 Licensee ID				X		

3.5.4.13.7 NCOALink options

This section describes the options in the NCOALink group. The related links list the sub option groups.

Option	Description
Mailing List Name	Enter the name of this list, up to 30 characters. If this list is a master house list or your only mailing list, consider entering your company name here. This name appears in the log files.
Platform ID	The platform ID is the NCOALink licensee's identification number that is assigned by the USPS. It's exactly four characters long.

Related Information

Processing options [page 1326]
Report Options [page 1328]
NCOALink Output Options [page 1329]
Processing Acknowledgment Form (PAF) Details [page 1329]
Service Provider Options [page 1330]
Contact Detail List [page 1332]

3.5.4.13.7.1 Processing options

The following table describes the *NCOALink Processing Options*.

[2] With NCOALink enabled

Option	Description
Consider Moves Within Months	Use this setting to ignore change-of-address data older than the specified number of months. For example, enter 12 to use change-of-address data that has a move-effective date within the last 12 months. If you are an end user or limited service provider, enter a value from 6 to 18. If you're a full service provider or using ANKLink, enter a value from 6 to 48. If the option is blank, the transform uses all available data based on your license. The default is blank.
External Processes Updating List	Indicate whether the list undergoes additional processing before or after the USA Regulatory Address Cleanse transform.
High Match Rate Expectancy	The USPS wants to distinguish between files that have a legitimate reason for a high percentage of NCOALink matches and files that are fraudulently used to create mover lists. Select *None* or leave blank if you don't expect a high match rate. This option provides legitimate reasons for a high match rate. *None:* Default. *ANKLink Processed List:* An ANKLink-processed file contains records for people who have moved but you don't yet have their new address. This option is available only to full service providers. *Stage File:* If you're performing Stage I or Stage II testing, ensure that the *List Processing Objective* is set to a Stage option. *Return Mail List:* A returned mail list file contains records for mail that was returned to sender.
List Processing Mode	*Change Of Address:* You're processing this job to update it with the latest address data. Default. *Statistics Only:* You're processing this job to analyze statistics such as the number of records in your list that have updated addresses and the number of moves of each type. When you choose this option, you do not receive move-updated addresses. *Return Codes Only:* You're processing this job for informational purposes. When you choose this option and post to the NCOALink_Return_Code or ANKLink_Return_Code output component, you can see the return codes, which further explain whether matching records were found in the NCOALink directories and why or why not. With this option, you do not receive move-updated addresses.
List Processing Objective	Specify your reason for using NCOALink: *Employee Training:* You're processing this file as part of employee training. *Internal Database Testing:* You're testing with a licensee-owned database. *Marketing:* You're testing with external customer lists.

Option	Description
	Normal: You're processing the mailing list to update it before a mailing. Default. *Stage I* and *Stage II:* You're testing the matching performance against a USPS test file. The USPS scores the Stage II test file. Choose Stage I or Stage II only if you are processing a USPS test file. *System Testing:* You're processing this file as part of system testing such as loading USPS file updates. Note When certifying for CASS and DSF2, you indicate the reason in the *Assignment Options* > *USPS Certification Testing Mode* option.
Processing First Class Mail *Processing Periodicals* *Processing Standard Mail* *Processing Package Service Mail*	Select the types of mail to process by selecting *Yes* or *No* for each option. *Processing First Class Mail* defaults to *Yes*; the others default to *No*.
Retrieve Move Types	Choose the types of moves to process: *Business:* Business moves only. *Individual:* Individual moves only. *Individual and Business* *Individual and Family* *Individual and Family and Business:* Default.

3.5.4.13.7.2 Report Options

There is one option in the *NCOALink Report Options* group.

Option	Description
Generate Return Code Descriptions	The NCOALink Processing Summary report always includes a brief summary of return codes, and you can include more detailed return code descriptions using this option. Return codes indicate whether a record was affected by a move, how the NCOALink match was made, or why a match could not be made. *Yes*: Includes the report codes in the NCOALink Processing Summary report.

Option	Description
	No: Default. Excludes report codes from the NCOALink Processing Summary report.

3.5.4.13.7.3 NCOALink Output Options

There is one option in the *NCOALink Output Options* group.

Option	Description
Apply Move to Standardized Fields	Component output fields are not affected by this option. *Yes*: Default. Data Services updates standardized fields to contain details about the address available through NCOALink. *No*: Standardized output fields have the standardized version of input rather than the moved address.

3.5.4.13.7.4 Processing Acknowledgment Form (PAF) Details

The following table describes the NCOALink PAF Details. PAF Details are not required for end users.

Option	Description
Company Website	Enter the company website address for the person signing the PAF. You can leave this parameter blank.
Customer Alternate Company Name	If the list owner's company is also known by another name, enter that alternate name here.
Customer Parent Company Name	If the list owner's company is owned by another company, enter the parent company's name here.
Date Signed By Licensee	Enter the date that the licensee (the NCOALink service provider) signed the PAF in yyyy/mm/dd format.
Date Signed By Customer	Enter the date the customer signed the PAF in yyyy/mm/dd format.
Email of Person Signing	Enter the email address for the person who is signing the PAF. You can leave this parameter blank.
Name Of Person Signing	Enter the name of the person signing this PAF, up to 50 characters.
Title Of Person Signing	Enter the job title of the person signing this PAF, up to 50 characters.

Option	Description
Type	*Initial*: This is the first PAF you're completing to become authorized to process addresses for this particular customer. *Modified*: You're completing a new PAF because some information on your old one changed. *Renewal*: You're completing a new PAF because your old one is expiring.
Using Alternative PAF	*Yes*: Select if you are using a PAF that is not the USPS form (you must have permission from the USPS). *No*: The default setting for this field. This field requires either a *Yes* or *No.*
Using Cooperative Database	Indicates whether the list is from a cooperative database. Applicable for Full and Limited Service Providers only. *Yes* *No:* (the default setting) Note A PAF must be on file for each participant in the cooperative database. Note When set to Yes, a "C" is included in the PAF log to indicate that the list processed was a cooperative database.

3.5.4.13.7.5 Service Provider Options

The following table describes the *NCOALink Service Provider Options*. These options are not required for end users.

Option	Description
Additional Notes	*None*: Default. *Customer Requested Extension*: Select if the customer submitted a written request for an extension.
Buyer Company Name	If the list was processed for rent, sale, or lease, enter the name of the company or individual who bought the list.
Concurrent Processed Data Modified	*No*: If you are processing this data in some other way while performing NCOALink processing, indicates that the concurrent processing does not include changes to the data. Default.

Option	Description
	From Postal Data: If you are processing this data in some other way while performing NCOALink processing, indicates whether the processing includes changes with postal data. *From Non Postal Data*: If you are processing this data in some other way while performing NCOALink processing, indicates whether the processing includes changes with non-postal data. *From Both*: If you are processing this data in some other way while performing NCOALink processing, indicates whether the processing includes changes with both postal and non-postal data.
Concurrent Processes Performed	Indicates whether you processed or will process this data in some other way while performing NCOALink processing. *Yes* *No*
In House List Processing	Indicates whether the list is an in-house (internal) list. Applicable for Full Service Providers only. *Yes* *No* Note When set to Yes, an “I” is included in the Customer Service log to indicate that the list was an in-house list.
Output Returned	Identifies the type of output returned to the client. *Standard*: All required NCOALink output was returned to the client. Default. *Modified*: One or more post processes modified the return information (updates were applied to the list). *Both*: One or more post processes modified the return information (updates were applied to the list); however, a separate file containing all of the required output was also returned.
Post Processed Data Modified	*No*: If you are processing this data after performing NCOALink processing, indicates that the postprocessing does not include changes to the data. Default. *From Postal Data*: If you are processing this data after performing NCOALink processing, indicates whether the postprocessing includes changes with postal data. *From Non Postal Data*: If you are processing this data after performing NCOALink processing, indicates whether the postprocessing includes changes with non-postal data.

Option	Description
	From Both: If you are processing this data after performing NCOALink processing, indicates that the postprocessing includes changes with both postal and non-postal data.
Post Processes Performed	Indicates whether you are processing this data after performing NCOALink processing. *Yes* *No*
Postcode For Mail Entry	Specifies the ZIP Code of the Business Mail Entry Unit (BMEU) or post office where the mail will be submitted for mailing.
Pre Processed Data Modified	*No*: If you are processing this data before performing NCOALink processing, indicates that the preprocessing does not include changes to the data. Default. *From Postal Data*: If you are processing this data before performing NCOALink processing, indicates whether the preprocessing includes changes with postal data. *From Non Postal Data*: If you are processing this data before performing NCOALink processing, indicates whether the preprocessing includes changes with non-postal data. *From Both*: If you are processing this data before performing NCOALink processing, indicates that the preprocessing includes changes with both postal and non-postal data.
Pre Processes Performed	Indicates whether you processed or will process this data before performing NCOALink processing. *Yes* *No*

3.5.4.13.7.6 Contact Detail List

The following table describes the *NCOALink Contact Details* options that are located in the Contact Detail List group. These options are not required for end users.

Option	Description
Address	Enter the broker's or list administrator's address.
Contact Level	Enter the degree of separation this contact is from you from 1 to 99. For example, enter 1 if you received the list from this contact. If your contact received the list from a different broker, then enter 2 for this contact.

Option	Description
	Note that the transform doesn't use this value in any logs.
Contact Company Website	Enter the website of the broker or list administrator. You can leave this parameter blank.
Locality	Enter the broker's or list administrator's locality (city).
Type	*Broker:* A broker directs business to the service provider. *List Administrator:* A list administrator stores and maintains address lists.
License Assigned ID	Enter a unique six-character ID number for the broker or list administrator. You assign the ID number.
NAICS Code	Enter the broker's or list administrator's numeric North American Industry Classification System code, which identifies the business in which they engage. For more information, see *http://www.census.gov/epcd/www/naics.html* .
Name	Enter the broker's or list administrator's name.
PAF Signing Date	Enter the date when the contact signed the PAF in the format yyyy/mm/dd.
Phone	Enter the broker's or list administrator's phone number.
Postcode1	Enter the broker's or list administrator's Postcode1 (ZIP Code).
Postcode2	Enter the broker's or list administrator's Postcode2 (ZIP+4 Code).
Region	Enter the broker's or list administrator's region (state).

3.5.4.13.8 Assignment options

With this option group, you can choose the add-on features that you want to use during processing.

Option	Description
Dual Address	Specifies the action to take when the transform encounters a dual address. *Position:* Selects an address based on the arrangement of the input data. The transform attempts to validate the address that is closest to the lower left corner of the address block. That might be the postal address (rural route or PO Box) or the street address; it depends on how the data was entered. *Postal:* The transform attempts to validate based on the postal address. If that fails, the transform attempts again based on the street address. *Street:* The transform attempts to validate based on the street address. If that fails, the transform attempts again based on the postal address.

Option	Description
Enable DPV	Specifies whether to perform DPV (Delivery Point Validation) processing. *Yes:* Enables DPV processing. *No:* Disables DPV processing.
Enable DSF2 Augment	Specifies whether to perform DSF2 (Delivery Sequence File Second Generation) augment processing. *Yes:* Enables DSF2 augment processing. *No:* Disables DSF2 augment processing.
Enable eLOT	Specifies whether to perform eLOT (Enhanced Line of Travel) processing. *Yes:* Enables eLOT processing. *No:* Disables eLOT processing.
Enable EWS	Specifies whether to perform EWS (Early Warning System) processing. If this transform cannot make an exact match within the `zip4us.dir` (Address Directory 1), it searches the EWS directory to see if the address is a new delivery point. If the address is located in the EWS directory, the transform marks the record as an EWS match and does not attempt further assignment. *Yes:* Enables EWS processing. *No:* Disables EWS processing.
Enable LACSLink	Specifies whether to perform LACSLink (Locatable Address Conversion System) processing. *Yes:* Enables LACSLink processing. *No:* Disables LACSLink processing.
Enable NCOALink	Specifies whether to perform NCOALink (National Change of Address) processing. *Yes:* Enables NCOALink processing. *No:* Disables NCOALink processing.
Enable RDI	Specifies whether to perform RDI (Residential Delivery Indicator) processing. *Yes:* Enables RDI processing. *No:* Disables RDI processing.
Enable Reverse Soundex Search	Specifies whether to use the `zip4us.rev` (Reverse Soundex) directory in an attempt to make address assignments. *Yes:* Enables Reverse Soundex. *No:* Disables Reverse Soundex.

Option	Description
Enable SuiteLink	Specifies whether to perform SuiteLink processing. *Yes:* Enables SuiteLink processing. *No:* Disables SuiteLink processing.
Geo Mode	Specifies which type of GeoCensus processing you want to perform. *Address:* The transform processes address-level GeoCensus only. *Both:* The transform makes an address-level GeoCensus assignment first if applicable. If no assignment is made, the transform makes a centroid-level GeoCensus assignment if applicable. *Centroid:* The transform processes centroid-level GeoCensus only. *None:* The transform turns off GeoCensus processing. Choose this option if you have not purchased the GeoCensus option or if you do not want to perform GeoCensus processing. i Note GeoCensus functionality in the USA Regulatory Address Cleanse transform will be deprecated in a future version. It is recommended that you upgrade any data flows that currently use the GeoCensus functionality to use the Geocoder transform. For instructions on upgrading from GeoCensus to the Geocoder transform, see the *Upgrade Guide*.
USPS Certification Testing Mode	Indicates the type of certification being processed so that the software checks for the appropriate settings and issues warnings and errors when applicable. *None:* The transform processes the job normally, without any special settings for certification. This is the default setting. *CASS:* The transform processes the job with the appropriate settings for CASS self-certification. *DSF2 Augment:* The transform processes the job with appropriate settings for DSF2 augment certification. i Note When certifying for NCOALink, you set the testing mode in the *NCOALink* > *Processing Options* > *List Processing Objective* option.

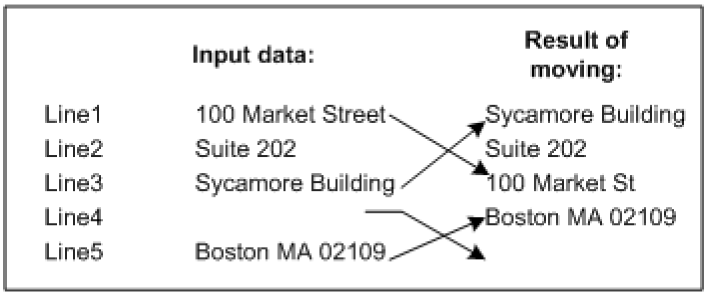
3.5.4.13.9 Standardization options

This option group contains all of the standardization settings that you need to define for processing USA data. (The options are listed alphabetically.)

Option	Description
Add Firm Match Secondary	*Yes*: Adds secondary address information obtained from SuiteLink directories to the address line. *No*: Does not add secondary address information obtained from SuiteLink directories to the address line, but includes SuiteLink-found information reflected in the lastline ZIP+4 Code and in other output fields. CASS users who do not want to update address lines in their data with SuiteLink secondary information should set this option to No. The software updates the last line to reflect the SuiteLink secondary information in the ZIP+4, and does not update the original address. The software also updates the address line based on your standardization settings in the job setup. Sometimes the option setting has no effect on the presence of the SuiteLink secondary address information. This can happen when the secondary address does not match the National directories, but exactly matches the SuiteLink secondary address information. In this situation, the software does not consider the SuiteLink secondary address as a change to the input secondary address, and does not remove the secondary address from the output address even when the Add Firm Match Secondary option is set to No. This applies to an exact match to the SuiteLink secondary unit and range, or, in cases where there is no unit designator on input, an exact match to the SuiteLink secondary range.
Address Line Alias	Specifies how to standardize the address line if the input primary address is an alias. *Convert*: Converts address lines to the preferred form found in the postal directory. *Preserve*: Retains address lines as they were input. Note To be compliant with CASS, set up your jobs to return the USPS preferred address. When the *Address Line Alias* option is set to *Convert*, the USPS preferred address is returned, even when the input record has a base address or an alias address. You can choose to set up your job to preserve the preferred address (*Address Line Alias* set to *Preserve*), but the software does not produce a USPS 3553 form.
Append Private Mailbox	Private mailboxes (PMB) are like post office boxes, except that they are hosted by private companies. *Yes*: Places the address and PMB in the same field. *No*: Places the PMB into a separate field. PMB information is output to the NON_POSTAL_SECONDARY_ADDRESS, NON_POSTAL_UNIT, and NON_POSTAL_UNIT_NUMBER fields.
Assign With Input Locality	Specifies whether to use the last-line index when assigning the locality (city) name.

Option	Description
	Yes: Assigns the Locality1 based on the locality name that is input if it is valid for the Postcode1. Does not change the Locality1 based on last line index. *No*: Assigns the Locality1 based on the locality that is input if it is valid for the Postcode1 and not a place name; otherwise, assigns Locality1 based on the last-line index of the address line. Produces a more geographically true Locality1. If you choose *No*, the value you choose for the *Preserve Place Names* option does not matter; place names are converted. Note When the *Use USPS Locality Abbreviation* option is set to *Yes*, *Preserve Place Names* is set to *Yes*, and *Assign with Input Locality* is set to *No*, the software may not preserve some place names over 13 characters and abbreviates them.
Capitalization	Specifies the casing of your address data. *Lower*: Converts data to all lowercase letters. For example, "Main Street South" becomes "main street south." *Mixed*: Converts data to initial capital letters. For example, "MAIN STREET SOUTH" becomes "Main Street South." *Upper*: Converts data to all capital letters. For example, "Main Street South" becomes "MAIN STREET SOUTH." Note If you want consistent casing for your data, make sure that this option and the *Capitalization* setting in the Data Cleanse transform are the same.
Combine Multilines	Specifies what to do with related fields input on separate lines. *Yes*: Looks for related fields that were input on separate lines, and tries to put them together on the same line. *No*: Does not try to combine fields.
Directional Style	Specifies whether to abbreviate directional data. *Long*: Uses fully-spelled directionals such as "North," "South," "East," and "West." *Preserve*: Preserves the style used in the input record. *Short*: Uses abbreviated directionals such as "N," "S," "E," and "W." Note When the *Use USPS Street Abbreviation* option is set to *Yes*, the software overrides a setting of *Long* and *Preserve* for the *Directional Style* option and outputs the short directional style.

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>*Include Unused Address Line Data*</td><td>
Specifies whether to output the unused address line data (for discrete and multiline fields).
Yes: Includes the unused address line data, including invalid secondary information, in the address line and in the ADDRESS_LINE_REMAINDER1 field.
No: Does not output the unused address line data in the address line but outputs the information to certain output fields.
Example:
<table>
<tr><th>Input</th><th colspan="2">Include Unused Address Line Data option</th></tr>
<tr><td></td><td>**Yes**</td><td>**No**</td></tr>
<tr><td>SAP
332 FRONT ST S FL 9
LA CROSSE WI 54601</td><td>SAP
332 FRONT ST S FL 3 FL 9
LA CROSSE WI 54601-4025</td><td>SAP
332 FRONT ST S FL 3
LA CROSSE WI 54601-4025</td></tr>
<tr><td colspan="3">*Output fields*</td></tr>
<tr><td>Primary_Secondary_Address</td><td>332 FRONT ST S FL 3</td><td>332 FRONT ST S FL 3</td></tr>
<tr><td>Full_Address</td><td>332 FRONT ST S FL 3 FL 9</td><td>332 FRONT ST S FL 3</td></tr>
<tr><td>Address_Line_Remainder1</td><td>FL 9</td><td>FL 9</td></tr>
<tr><td>Secondary_Address</td><td>FL 3</td><td>FL 3</td></tr>
<tr><td>Pre_SuiteLink_Unit_Description</td><td>FL</td><td>FL</td></tr>
<tr><td>Pre_SuiteLink_Unit_Number</td><td>9</td><td>9</td></tr>
</table>
</td></tr>
<tr><td>*Move Multiline Data*</td><td>
If you turn on this feature, the *USA Regulatory Address Cleanse* transform rearranges your multiline data to conform to USPS guidelines.
The transform moves the primary address into position above the locality-region-postal code line (or lastline).
Bottom: Rearranges the lines according to USPS guidelines. If there are any blank lines, the transform moves them to the top and shifts the data to the bottom of the block.
No: Does not rearrange any lines, blank or otherwise.
Top: Rearranges the lines according to USPS guidelines. If there are any blank lines, the transform moves them to the bottom and shifts the data to the top of the block.
Example:
</td></tr>
</table>

Option	Description
	 Input data: Result of moving: Line1 Line2 Line3 Line4 Line5 100 Market Street Suite 202 Sycamore Building Boston MA 02109 Sycamore Building Suite 202 100 Market St Boston MA 02109 This feature does not require that you standardize your data. *Note:* If you choose *Top* or *Bottom*, the input field lengths of all fields mapped to Multiline**<x>** must be the same. For example, if Multiline1 is set to a length of 60, all Multiline fields that you use must be set to 60.
Multiline Update Postcode 1	This option affects multiline data only (data passed in and retrieved through multiline fields). *Dont_Update*: Assigns Postcode1 fields, but does not write them to the Multiline output fields. In those fields, leaves the original Postcode1 intact and the assigned Postcode1 is available in other output fields. *Erase_Then_Update*: Replaces the original Postcode1 with the assigned Postcode1 in the Multiline output fields. If no Postcode1 is assigned, the original Postcode1 is not included in the Multiline output fields. *Update*: Replaces the input Postcode1 with the assigned Postcode1 in the Multiline output fields. If no Postcode1 is assigned, the original is retained.
Multiline Update Postcode 2	This option affects multiline data only (data passed in and retrieved through multiline fields). *Dont_Update*: Assigns Postcode2, but does not write them to the Multiline output fields. The transform leaves the original Postcode2 intact and the assigned Postcode2 is available in other output fields. *Erase_Then_Update*: In the Multiline output fields, replaces the original Postcode2 with the assigned Postcode2. If no Postcode2 is assigned, the original Postcode2 is not available in the Multiline output fields. *Update*: In the Multiline output fields, replaces the input Postcode2 with the assigned Postcode2. If no Postcode2 is assigned, it retains the original.
Preserve Dual Address Order	Specifies whether to preserve or change the dual address order. *Yes*: When an address contains both a street and mailing address, keeps the address order as it was input. *No*: When the input address contains both a locality and mailing address, moves the assigned address immediately above the locality and region.
Preserve Place Names	Specifies whether to preserve or change non-mailing city names (place names). *Yes*: Preserves the non-mailing city name. Given Hollywood as input, the transform outputs Hollywood.

Option	Description
	No: Changes non-mailing city names to city names preferred by the USPS. Given Hollywood as input, the transform outputs Los Angeles. ℹ Note When the *Use USPS Locality Abbreviation* option is set to *Yes*, *Preserve Place Names* is set to *Yes*, and *Assign with Input Locality* is set to *No*, the software may not preserve some place names over 13 characters and abbreviates them.
Primary Type Style	Specifies whether to abbreviate the street (primary) type. *Long:*: Uses fully-spelled primary types such as Street, Avenue, and Road. *Preserve*: Preserves the style used in the input record. *Short*: Uses abbreviated primary types such as St, Ave, and Rd. ℹ Note When *Use USPS Street Abbreviation* is set to *Yes*, the software overrides a setting of *Long* and *Preserve* for the *Primary Type Style* option and outputs the short primary type style.
Retain Pound Sign in Unit Description	Outputs "#" into either extraneous fields or to the UNIT_DESCRIPTION output field. *Yes*: Outputs the # unit designator to the UNIT_DESCRIPTION output field. *No*: Outputs the # unit designator to the EXTRANEOUS_SECONDARY_UNIT_NUMBER and/or the EXTRANEOUS_SECONDARY_ADDRESS_DATA output fields. The option does not affect the following address situations: ● Puerto Rican addresses ● Military addresses ● Rural Route addresses ● Addresses without "#" in the address line ● Addresses with remainder words
Standardize Assigned Address	Specifies whether to correct and standardize the assigned address line and lastline data. *Yes*: Corrects and standardizes your address line and lastline data. Use this value for CASS certification. *No*: Does not standardize your address line or lastline data.
Standardize Unassigned Address	Specifies whether to standardize unassigned data. *Yes*: Attempts to parse and standardize any unassigned addresses. *No*: Leaves unassigned addresses as entered on input.
Unit Description	Specifies how to standardize the unit description. *Convert*: Uses the unit description found in the postal directory (such as an apartment, suite, room, or floor).

Option	Description
	Preserve: Preserves the unit description from the input record and corrects any spelling errors.
Use USPS Locality Abbreviation	*Yes*: Enables this option (and affects only the multiline and standardized last line fields). *No*: Disables this option. Provides a USPS 13-character city name when one is available. If the city name is not valid (for example, it is a non mailing city name), the software relies on other settings in the job to determine what to output for city. If the city name is longer than 13 characters, the software returns an abbreviation that is 13 characters or less. If the city name is already 13 characters or less, the software does not abbreviate it. Note When the number of characters in the output is greater than the length specified for the output field, the software attempts to truncate the output data to fit in the output field without eliminating vital address data. Intelligent truncation abbreviates the output data first, and if it still doesn’t fit the output buffer, it truncates the data. There are no options to set this up in the USA Regulatory Address Cleanse transform. The transform does this automatically. If the *Use USPS Primary Name Abbreviation* and/or the *Use USPS Locality Abbreviation* options are enabled, the software uses those abbreviations first. If the values don't fit within the length of the output fields, then intelligent truncation occurs.
Use USPS Primary Name Abbreviation	*Yes*: Enables this option. *No*: Disables this option. Abbreviates the address line to 30 characters or less when the output address line exceeds 30 characters and when an abbreviated form of the address is available in the directory data supplied by the USPS. Abbreviated forms of an address are only provided for output addresses 30 characters or greater. If the output address line is already 30 characters or less, the output address line is not abbreviated. Note An address line may be output with more than 30 characters in situations where no abbreviated form of the address is available in the directory data. If your data must fit exactly into 30 characters, we recommend to set appropriate address output field lengths to 30. This option affects multiline and standardized last line fields when set to *Yes*. If the *Use USPS Street Abbreviation* option is set to *Yes*, it affects the following address components on output: ● *Suffix Style*: The style will be short.

Option	Description
	• *Directional Style*: The style will be short. • *Address Line Alias*: The setting of *Preserve* may be overridden. Note When the number of characters in the output is greater than the length specified for the output field, the software attempts to truncate the output data to fit in the output field without eliminating vital address data. Intelligent truncation abbreviates the output data first, and if it still doesn't fit the output buffer, it truncates the data. There are no options to set this up in the USA Regulatory Address Cleanse transform. The transform does this automatically. If the *Use USPS Primary Name Abbreviation* and/or the *Use USPS Locality Abbreviation* options are enabled, the software uses those abbreviations first. If the values don't fit within the length of the output fields, then intelligent truncation occurs.

3.5.4.13.10 Non Certified options

This option group includes options to process your data without following CASS certification rules.

Option	Description
Accept Inexact Postcode Move	When an input record has an obsolete postcode or a postcode move, specifies whether the transform should ignore some of the non-matching elements between the input record and the record in the national directory, and use built-in matching thresholds to determine if the records match. *Yes*: Ignores the non-matching elements and uses built-in matching thresholds. *No*: Disables this option (and does not ignore non-matching elements).
Assign With Input Postcode	Specifies whether the transform should use the last four digits of the 9-digit postcode (if present on input), which is usually the Postcode2 field (ZIP+4), during address assignment. *Yes*: Enables the transform to use the record's last four digits of the 9-digit postcode to try to make a finer level of assignment than it could make under CASS rules. Under CASS rules, the transform doesn't consider the last four digits of the 9-digit postcode. In order for this option to work, the last four digits of the 9-digit postcode must be unique to a valid firm or secondary address. *No*: Disables this option.

Option	Description
Assign Postcode2 Not DPV Validated	Specifies whether to output the Postcode2 when an assignment is made even when *Enable DPV* is set to *No* and *Disable Certification* is set to *Yes*. The output address is not validated by DPV. *Yes*: Assigns the Postcode2 when *Enable DPV* is set to *No*. *No*: Leaves the Postcode2 blank when an assignment is made and *Enable DPV* is set to *No*.
Disable Certification	Specifies whether to run address cleansing for CASS certification. *Yes*: Runs address cleansing without the restrictions of CASS certification rules. Choose this value if you want to use any of the other options in this option group. This value also enables non-mailers to process addresses for 14 months after the directory creation date rather than 3 to 4 months for postal discounts through the USPS. You will not receive any postal discounts with this value or be able to produce a USPS Form 3553. *No*: Runs address cleansing under CASS certification rules. The other options in this option group are ignored.
Enable Geo Only	ℹ Note GeoCensus functionality in the USA Regulatory Address Cleanse transform will be deprecated in a future version. It is recommended that you upgrade any data flows that currently use the GeoCensus functionality to use the Geocoder transform. For instructions on upgrading from GeoCensus to the Geocoder transform, see the *Upgrade Guide*. Specifies whether the transform should process only with the address-level GeoCensus or centroid GeoCensus directories. Choose the type of GeoCensus processing in the Assignment Options group. *Yes*: Runs address cleansing with the GeoCensus directories only. The transform does not process your data with the postal directories. Make sure that you have defined the location of the GeoCensus directories in the Reference Files option group. You must also include the appropriate GeoCensus output fields; data is not posted in any other output fields. *No*: Enables you to run address cleansing with the GeoCensus directories, as well as the postal directories.
Enable Parse Only	Specifies whether the transform should parse and validate your data or parse only. *Yes*: Parses records into their discrete components, but does not perform a lookup in the postal directories. This mode is fast, but parsing results are unverified. *No*: Parses records into their discrete components and performs a lookup in the postal directories. This mode may be slower, but parsing results are verified.
Enable Suggestion Lists	Specifies whether suggestion lists are generated for records that cannot be assigned. This option is for transactional projects.

Option	Description
	Yes: Generates suggestion lists. *No*: Does not generate suggestion lists.

3.5.4.13.11 CASS Report options

With this option group, you add the necessary USPS Form 3553 information as required by the USPS when certifying a mailing.

Option	Description
Company Name Certified	If you rely on SAP for vendor CASS certification, leave this parameter blank; the transform inserts "SAP" as the default value. If you have your own end-user CASS certification from the USPS, type your company name (up to 40 characters).
List Name	Specifies the name of the mailing list (up to 20 characters).
List Owner	Specifies the name of your company (up to 19 characters).
LOT Certification	Specifies whether you have LOT certification. *Yes:* You have LOT certification but you do not have CASS certification in your own name. *No:* You have CASS certification in your own name but you did not seek or obtain LOT certification. In this case, setting LOT Certified to No ensures that the LOT Certification lines on your USPS 3553 forms are blank, which is appropriate.
Mailer Address 1	Specifies the name and address of the person or organization for whom you are preparing the mailing (up to 29 characters per line).
Mailer Address 2	
Mailer Address 3	
Mailer Address 4	
Software Version	If you rely on SAP for vendor CASS certification, you may leave this parameter blank. The transform inserts the appropriate software name and version as the default value. If you have received end-user CASS certification in your own company's name, type the software name and version number that you use to receive CASS certification.

3.5.4.13.12 Suggestion List group

Set the options in this group to configure how suggestion lists are output.

Option/Option group	Description
Address Lines Match Minimum	Specifies the similarity score required for address-line suggestions. Valid values are 0 to 80. The similarity score determines which suggestions are returned in the list. A higher number indicates that the suggestion must be more similar to the input to be returned as a possible suggestion.
Address Range Window	Specifies a number that represents a span. The software uses the number to present a range of addresses around the input primary address range for which to return suggestions. By using this option, you can limit the suggestions returned to be within a few blocks of your input. For example, assume you entered 500 for this value. Then, you submit the following street address: 1000 Pine St. Suggestions would only be returned in a range from 750 to 1250 Pine Street. Type "0" if you don't want to limit the ranges returned in suggestions.
Combine Overlapping Ranges	Specifies how individual suggestions with overlapping ranges are consolidated. *Combine_Ignoring_Gaps:* Ignores gaps and overlaps in primary ranges, so consolidation is more aggressive. *Combine_Preserving_Gaps:* Preserves gaps in primary ranges, but overlapping ranges are consolidated. *None:* Suggestions are not consolidated.
Delimiter	Specifies the delimiter to use between each suggestion. This is applicable if you chose *Delimited* for the *Style* option. Choose any character or string to separate each suggestion. This value should differ from the *Field Delimiter* value.
Field Delimiter	Specifies the delimiter to use between each suggestion list. This is applicable if you chose *Delimited* for the *Style* option. This value should differ from the *Delimiter* value.
Lastlines Match Minimum	Specifies the similarity score required for lastline suggestions. Enter a value from 0 to 80.

Option/Option group	Description
	The similarity score determines which suggestions are returned in the list. A higher number indicates that the suggestion must be more similar to the input to be returned as a possible suggestion.
Match Range	Specifies whether to disregard an address-line suggestion when it does not match the primary range of the input address. *Yes:* Returns address-line suggestions only when they match the primary range of the input address. *No:* Returns a possible address-line suggestion when it doesn't have the same primary range as the input.
Max Number Addresslines	Specifies the maximum number of address-line suggestions that can be generated. The maximum number that you can enter is 100. For example, you could set this option to limit the size of the SOAP documents being sent by the web service, or to limit the maximum number of suggestions that your users would have to choose from. ℹ Note If you set a low maximum, a viable suggestion could be left out of the suggestion list.
Max Number Lastlines	Specifies the maximum number of lastline suggestions that can be generated. The maximum number that you can enter is 15. For example, you could set this option to limit the size of the SOAP documents being sent by the web service, or to limit the maximum number of suggestions that your users would have to choose from. ℹ Note If you set a low maximum, a viable suggestion could be left out of the suggestion list.
Style	Specifies the style of the output file. *Delimited:* Outputs the suggestion list data in a delimited text format, with the delimiters specified in the *Delimiter* and *Field Delimiter* options. *XML:* Outputs the suggestion list data as hierarchical XML. This option is likely the preferred one for users who integrate suggestion lists via the web service. You can then use the XML tools that you own to parse the suggestion list data.
Suggestion List Components	Specifies the address field components that you want to include in the Suggestion_List output field.

Option/Option group	Description
	ℹ Note Suggestion list field components that do not have a value are not output to the Suggestion_List output field if the selected *Style* is *XML*.

Related Information

Suggestion List Components [page 1347]
Designer Guide: Data Quality, Similarity score [page 548]
Designer Guide: Nested Data, Extracting data quality XML strings using extract_from_xml function [page 370]

3.5.4.13.13 Suggestion List Components

These options let you choose what information to output to the Suggestion_List output field.

> ℹ Note
>
> If XML is the selected Style in the Suggestion List option group, suggestion list fields that do not have a value are not output to the Suggestion_List output field.

Option/option group	Description
Firm	Select Yes to output the firm name for the secondary address.
Locality1	Select Yes to output the locality1 (city) preferred by the USPS. Applicable for primary, secondary, and lastline address levels.
Postcode	Select Yes to output the five-digit Postcode1 (not including the four-digit ZIP4). Applicable for primary, secondary, and lastline address levels.
Postcode2 Even	Select Yes to output the four-digit ZIP4 code, even numbers only. Applicable for primary and secondary address levels.
Postcode2 Odd	Select Yes to output the four-digit ZIP4 code, odd numbers only. Applicable for primary and secondary address levels.
Primary Name1	Select Yes to output the street name description. Applicable for primary and secondary address levels.

Option/option group	Description
	ℹ Note Primary numbers (high or low) are not output to this field.
Primary Name Full1	Select Yes to output the primary address line, such as the street address or post office box. The output includes the following information: Primary Prefix1, Primary Name1, Primary Type1, and Primary Postfix1. Applicable for primary and secondary address levels. ℹ Note Primary numbers (high or low) are not output to this field.
Primary Number High	Select Yes to output the high portion of the premise number range. Applicable for primary and secondary address levels.
Primary Number Low	Select Yes to output the low portion of the premise number range. Applicable for primary and secondary address levels.
Primary Postfix1	Select Yes to output the abbreviated directional (for example, N, S, NW, or SE) that follows a street name. Applicable for primary and secondary address levels.
Primary Prefix1	Select Yes to output the abbreviated directional (for example, N, S, NW, or SE) that precedes a street name. Applicable for primary and secondary address levels.
Primary Side Indicator	Select Yes to output Odd or Even for the primary side indicator. Applicable for primary and secondary address levels.
Primary Type1	Select Yes to output the abbreviated street type (for example, St, Ave, or Pl). Applicable for primary and secondary address levels.
Region1	Select Yes to output the state, province, territory, or region. Applicable for primary, secondary, and lastline address levels.
Secondary Side Indicator	Select Yes to output Odd or Even for the secondary side indicator. Applicable for secondary address level.
Selection	Select Yes to output the selection number for multiple suggestions.
Unit Description	Select Yes to output the unit description (for example, #, Apartment, or Flat). Applicable for secondary address level.
Unit Number High	Select Yes to output the high portion of the unit number range. Applicable for secondary address level.
Unit Number Low	Select Yes to output the low portion of the unit number range. Applicable for secondary address level.

3.5.4.13.14 Z4 Change options

With the Z4Change options you can turn on Z4Change processing and specify the last time the Postcode2 was updated.

Option	Description
Enable Z4 Change	Specifies whether to enable Z4Change processing. *Yes:* Turns on Z4Change processing. *No:* Turns off Z4Change processing.
Last ZIP4 Assign Date	Specifies the month and year that the input records were most recently ZIP+4 coded—either through a full address correction process or a previous Z4Change pass. Enter the date using the format MM/YYYY. For example, enter a date of January 2004 by typing **`01/2004`**. The USA Regulatory Address Cleanse transform verifies that your date is within the 12-month period covered by the Z4Change file. If there is a date problem, you will receive an error message when Data Quality runs your project.

3.5.4.13.15 USA Regulatory Address Cleanse fields

The USA Regulatory Address Cleanse transform requires that you map fields on input and output.

Related Information

Input fields [page 1351]
Output fields [page 1353]

3.5.4.13.15.1 Field category columns in Output tab

The Output tab lists output fields that hold the data that the transform cleanses or creates. You can choose to view the Best Practice, In Use, or All output fields by selecting the corresponding option at the top of the tab.

Best Practice: Lists all available output fields that have a field class of Best.

In Use: Lists only the output fields that you have chosen to output (listed in Schema Out).

All: Lists all output fields that are available for this transform.

> Note
>
> For details about mapping input and output fields, see the *Designer Guide*.

The output field attributes in the table below are listed in groups based on the field category column. Each field has categories that describe the type of content that is output. The field category displays “None” when it does not apply to the field.

Category	Description
Content Type	Identifies the type of data in the field. Setting the content type helps you map your fields when you set downstream transforms.
Field Addrclass	Specifies the address class for the generated field. *Delivery:* When used with the applicable Field Name, this value generates fields that reflect the address that is used in an attempt to assign an address. *Dual:* When used with the applicable Field Name, this value generates fields that reflect the address that is not used in an attempt to assign an address for input records that may contain both a street and postal address on input. *Official:* When used with the applicable Field Name, this value generates fields in the form of the data preferred by the Postal Authority. For example, in Winona, Minnesota USA, Broadway and 6th Street are alternate names for the same street. A letter addressed to Broadway is delivered, but the USPS prefers 6th Street.
Field Category	*Component:* Individual address components and postal codes that are related to the processed record. *Standardized:* Standardized input lines based on the settings in the Standardization Options group in the transform. *Suggestion:* Suggestion list output data based on the settings in the Suggestion List Options group.
Field Class (USA Regulatory Address Cleanse)	Specifies the field class that you want to assign to your output fields. *Best:* Outputs data based on various factors, such as whether an address was assigned, the Field AddrClass, and any settings that you defined in the Standardization Options group in the Options tab. Note When NCOALink is enabled and a valid move is available, Best fields contain the move-updated address data if it exists and if it matches in the U.S. National Directories. Or, the field contains the original address data if a move does not exist or if the move does not match in the U.S. National Directories. *Correct:* Outputs the complete and correct value found in the directories, and is standardized according to any settings that you defined in the Standardization Options group in the Options tab. *Parsed:* Outputs the parsed value.

Category	Description
	Pre_LACSLink: Retained address components that were replaced with LACSLink address information. *Move_Updated*: Outputs the address components that have been updated with move-updated address data. Note The transform looks for the move-updated address information in the U.S. National Directories. When the move-updated address is not found in the U.S. National Directories, the software populates the Move Updated fields with information found in the Move Update Directories only. The Move Updated fields that are populated as a result of standardizing against the U.S. National Directories is not updated.
Field Class (Global Address Cleanse)	Specifies the field class that you want to assign to your output fields. *Best:* Outputs data based on various factors, such as whether an address was assigned, the Field AddrClass, and any settings that you defined in the Standardization Options group in the Options tab. *Parsed:* Outputs the parsed value.
Field Name	Specifies a field name where the data is populated based on the options that you specify within this transform.
Type	Specifies the type and default length of data the output field contains; for example, varchar, date, and time.

Related Information

Content types [page 1131]

Designer Guide: How address cleanse works [page 586]

3.5.4.13.15.2 Input fields

The following are Data Services input fields that you can use to map the input data file fields for the USA Regulatory Address Cleanse transform.

Field	Description
Address_Line	The delivery address line (for example, "123 Main Street, Unit 4").

Field	Description
Check_Digit	The check-digit for 11-digit delivery-point bar code. Applicable only if the transform can make a full assignment. The transform provides the check digit for a five-digit bar code when a five-digit assignment is possible, or the address is undeliverable. When the address is unassigned, the check digit is based on the unverified input Postcode1 (ZIP Code).
Country	The country name. This transform does not attempt to make an assignment for addresses outside of the U.S. and its possessions, territories, and protectorates.
County_Code	The three-digit county code. Numbers start at 001 within each state.
Data_Source_ID	The input source or list identifier. Use this field to identify the source of an input set or to identify the list that an input record belongs to in the case that multiple lists are present in the input. Statistics are generated for each unique value in this field when you map the field in conjunction with enabling the *Gather Statistics Per Data Source* option in the Reports and Statistics group.
Delivery_Point	The two-digit DPBC code.
Family_Name1	The family name (for example, Smith).
Firm	The company name.
Given_Name1	The given name (for example, Robert).
Given_Name2	The second given name (for example, B.).
Lastline	The last line delivery information that can include all or some of the following fields: Locality1, Region1, Postcode1, or Postcode2.
Locality1	The city, town, or suburb.
Locality2	The Puerto Rican urbanization information.
LOT	The Line-of-Travel number.
LOT_Order	The Line-of-Travel sortation: *A*: Ascending *D*: Descending LOT codes are required for non-automated, CART presorting in Standard Mail, Enhanced Carrier Route Subclass.

Field	Description
Multiline1-12	A line from the input file which may contain data. The type of data in this line may vary from record to record.
Name	The name of the person associated with the address.
Postcode_Full	The complete postal code (ZIP10 with a hyphen; ZIP9 without a hyphen).
Postcode1	The five-digit primary ZIP Code. It does not include the four-digit ZIP4 Code.
Postcode2	The four-digit ZIP4 code. On a mail piece, this code follows the primary postal code with a hyphen placed between; for example, 54601-1234.
Postname	The honorary postname (indicating certification, academic degree, or affiliation such as CPA) or maturity postname (indicating heritage such as Jr.).
Prename	The prename (for example, Mr.).
Region1	The name of the state or province for this address.
SortCode_Route	The four-digit carrier route number.
Stage_Address_Flag Stage_Lastline_Flag Stage_Name_Flag Stage_Record_Key	The USA Regulatory Address Cleanse information required from the stage file. For NCOALink stage testing only.
Suggestion_Reply1-5	The index number that corresponds to a specific lastline suggestion or an address line suggestion. These fields can also be used to input a street primary range or a street secondary range. If you do not want to use a suggestion list, make the value of this field 0, and the suggestion list will be ignored.
Unit_Number	The secondary address information (for example, the unit description and/or secondary number).

3.5.4.13.15.3 Output fields

The following are Data Services output fields that can be used for the USA Regulatory Address Cleanse transform. See the fields listed in the transform's Output tab to view each field's properties.

Field	Description
Address_Line	Complete, standardized primary and secondary address line. The style of suffixes, directional, and unit designators depends on how you define your options.

Field	Description
	i Note If the output values don't fit within the length of the output field, then intelligent truncation occurs.
Address_Line_Remainder1	Extraneous data found on the address line that cannot be identified as address data or does not belong in the standardized address line. This may include invalid secondary address information.
Address_Type (DELIVERY, DUAL)	The record-type indicator for the assigned address. Applicable for DELIVERY and DUAL Generated Field Address Class. The first character indicates the type of record in the address directory to which the address matched: *F*: Firm *G*: General delivery *H*: High-rise apartment or office building *M*: Military *P*: Post office box *R*: Rural route or highway contract *S*: Street (usually, one side of one city block) **`<blank>`**: Unassigned The second character may be a *D* or blank. The D stands for default; it means that the transform detected, from the address directory, that a finer level of address assignment would be possible if further input information were available. *FD*: Firm default. The transform did not assign a firm-level Postcode2, but could do so if given more or better firm information. *GD*: General delivery default. Assigned when General Delivery is the only primary name listed for the Postcode1. *HD*: High-rise default. The transform assigned the Postcode2 for the entire building. Assignment at the unit, floor, or wing level is possible. Often caused by a suite or apartment number out of range. *RD*: Rural route or highway contract default. The transform assigned the Postcode2 for the entire route but could make a better assignment with the box number. *SD*: Street default. Usually means that there is no Postcode2 for the building, so the transform had to assign the Postcode2 for the block. *UD*: Unique default. Either the owner of the unique Postcode1 has not provided Postcode2 assignments, or the address could not be matched.

Field	Description
	When the transform cannot assign an address, it provides an address-type indication based on the way that the input data was parsed. This process is not foolproof. The transform may indicate that a street, rural route, highway contract, general delivery, or PO Box was parsed.
AGeo_Countycode	Federal Information Processing Standard (FIPS) county code (for example, 063).
AGeo_Latitude	Latitude (degrees north of the equator) in the format 12.123456.
AGeo_Longitude	Longitude (degrees west of the Greenwich Meridian) in the format -12.123456.
AGeo_MCDCode	U.S. Census Bureau Minor Civil Division (MCD) data or if MCD data is unavailable, Census County Division (CCD) data (for example, 40775).
AGeo_Placecode	Federal Information Processing Standard (FIPS) place code (for example, 40775).
AGeo_Sectioncode	U.S. census tract code in the format 1234567890 (for example, 0003001059).
AGeo_Statecode	Federal Information Processing Standard (FIPS) state code (for example, 55).
Alias_Type (OFFICIAL)	The alias-type indicator for the assigned address. Applicable for OFFICIAL Generated Field Address Class. It describes the input address, not the output address. *A*: The input address matches an abbreviated street name. *B*: The input address matches the high-rise alternate default base record. *C*: The input street name is out of date; to get new street name, convert your record to the preferred alias. *H*: The input address is an undesirable alternate, subject to conversion to a USPS preferred street address (high-rise alternate). *O*: The input address is a street nickname or other alias. *P*: The input address is a preferred alias. `<blank>`: The input address is not an alias or is unassigned.
ANKLink_Return_Code	ANKLink return code (Attempted Not Known). Valid values are: *77*: An ANKLink match was found. If NCOALink_Return_Code contains an A, 91, or 92, you may be able to obtain a new address from an NCOALink full service provider. `<blank>`: No NCOALink lookup or no ANKLink match. This is always blank for full service providers.
Audit_Dropped_Secondary	This field is used for audit testing. This field is also populated when an ANKLink match is made.

Field	Description
Audit _Prename Audit _Given_Name1 Audit _Given _Name2 Audit_Family_Name1 Audit _Postname	These fields contain the name data used to make an NCOALink match. In some cases, the name in these fields is not the same as the input name (for example, if a nickname, alternate spelling, or initial is used instead). In the case of a firm match, these name fields contain a split version of the firm data. These fields are also populated when an ANKLink match is made.
Audit _Gender	This field is used for audit testing. This field is also populated when an ANKLink match is made.
Audit _General	This field contains information for Stage I and Stage II tests, specifically query data, result data, and hint bytes, as the USPS requires. Use this field for audit purposes only. This field is required for audits. For more information about the content of this field, see the NCOALink User Technical Reference at *http://ribbs.usps.gov/* . This field can also contain ANKLink return codes.
Audit _Primary_Name	This is the primary name that is sent to NCOALink for matching. This field is required for audits.
Audit _Range	This is the range that is sent to NCOALink for matching. This field is required for audits.
Audit _Secondary_Range	This is the secondary range that is sent to NCOALink for matching. This field is required for audits.
Audit _Truncated_Given_Name1	This field is used for audit testing. This field is also populated when an ANKLink match is made.
Audit _Truncated_Given_Name2	This field contains the truncated middle name as stored in the NCOALink data. Use this field for audit purposes only. This field is required for audits.
Audit_Unit	This is the unit data that is sent to NCOALink for matching. This field is required for audits.
Carrier_Route_Sort_Zone	The carrier-route sort zone indicates eligibility for Standard Mail Automation Enhanced Carrier Route. *A*: Carrier route rates are available and merging is allowed. *B*: Carrier route rates are available and merging is not allowed. *C*: Carrier route rates are not available and merging is allowed. *D*: Carrier route rates are not available and merging is not allowed.
CASS_Assignment_Type	Indicates the option used in making the assignment:

Field	Description
	O: The non-CASS and DPV tie-break options are disabled or not used to make an assignment. *1*: Inexact Postcode1 move assignment. *2*: Input Postcode2 assignment. *3*: DPV tie-breaking was used to make this assignment. **`<blank>`**: The transform cannot assign an input address.
CASS_Record_Type	The record type necessary for posting on the CASS test. This field is populated for assigned records only. The valid record types include: *F*: Firm *G*: General delivery *H*: High-rise *P*: Post office box *R*: Rural route or highway contract *S*: Street
CGeo_BSACode	A Core-Based Statistical Area (CBSA) that consists of: ● A county with an incorporated place or a census-designated place that has a population of at least 10,000. ● Adjacent counties with at least 25 percent of employed residents of the county who work in the CBSA's core or central county. For example, 29100. CBSAs are either metropolitan (population of at least 50,000) or micropolitan (population between 10,000 and 50,000). With CBSAs, you can collect statistics for less urban areas of the country. CBSAs cover approximately 90 percent of the entire U.S. population.
CGeo_Latitude	Latitude (degrees north of the equator) in the format 12.123456.
CGeo_Longitude	Longitude (degrees west of the Greenwich Meridian) in the format -12.123456.
CGeo_Metrocode	Metropolitan Statistical Area (MSA) number (for example, 3870). The value 0000 indicates that the address does not lie in any MSA (usually a rural area).
CGeo_Sectioncode	U.S. census tract and block group code (for example, 0202001067). The first six digits are the tract number, and the first of the final four digits is the block group code within the tract. The Metropolitan Statistical Area (MSA) and block group codes are used for matching to demographic-coding databases. To uniquely specify a census block group within the entire country, combine the Sortcode_Postcode and CGeo_Sectioncode fields.

Field	Description
Check_Digit	Check digit for the delivery-point bar code, or for a five-digit bar code if a full postal code (ZIP+4) could not be assigned.
Count	Specifies the suggestion count generated as the result of looking up the current record. A nonnegative value is output. If the current record does not end processing with a suggestion list needing resolution, then the value in this field is 0.
Country	The country name.
County_Code	Federal Information Processing Standard (FIPS) three-digit county code. Numbers are unique within states. You might use county information if you are preparing a presorted periodicals mailing.
County_Name	The fully-spelled county name.
Delivery_Point	The two-digit DPBC code.
Delivery_Type	Type of postal facility: *A*: Airport Mail Facility (AMF) *B*: Branch Office *C*: Community Post Office (CPO) *D*: Area Distribution Center (ADC) *E*: Sectional Center Facility (SCF) *F*: Delivery Distribution *G*: General Mail Facility (GMF) *K*: Network Distribution Centers (NDC) *M*: Money Order Unit *N*: City/place name *P*: Post Office (main) *S*: Station *U*: Urbanization (Puerto Rico only)
District	District number for the U.S. House of Representatives.
DPV_CMRA	The DPV Commercial Mail Receiving Agencies (CMRA) component that is generated for this record. *L*: The address triggered DPV locking. *N*: The address is not a CMRA. *Y*: The address is a valid CMRA.

Field	Description
	<blank>: A blank output value indicates that Enable_DPV_Validation is set to No, DPV processing is currently locked, or the transform cannot assign the input address.
DPV_Footnote	DPV footnotes are required for end-user CASS certification. The footnotes contain the following information: *AA*: The input address matches to the ZIP+4 file. *A1*: The input address does not match to the ZIP+4 file. *BB*: All input address field values match to DPV. *CC*: The input address primary number matches to DPV, but the secondary number does not match (the secondary is present but invalid). *F1*: The input address matches to a military address. *G1*: The input address matches a general delivery address. *M1*: The input address primary number is missing. *M3*: The input address primary number is invalid. *N1*: The input address primary number matches to DPV but the address is missing the secondary number. *NL*: An NCOALink move address cannot be DPV confirmed. The NCOALink directory data does not exactly match the DPV directory data. This may happen because the NCOALink directories are updated more frequently than the DPV directories. Note The NL footnote is applicable only for the Move Updated generated field class. *P1*: The input address is missing the rural route or highway contract box number. *P3*: The input address is an invalid post office, rural route, or highway contract number. *RR*: The input address matches to CMRA. *R1*: The input address matches to CMRA, but the secondary number is not present. *U1*: The input address matches a unique address. Note The transform always posts the DPV footers in the same order and this field is not always 12 characters in length.

Field	Description
DPV_NoStats	No Stat indicator. No Stat means that the address is a vacant property, it receives mail as a part of a drop, or it does not have an established delivery yet. *Y*: The address is flagged as No Stat in DPV data. *N*: The address is not No Stat. **`<blank>`**: The address was not looked up. ℹ Note The US Addressing report contains DPV NoStats counts in the DPV Summary section.
DPV_Status	The DPV status component that is generated for this record. *D*: The primary range is a confirmed delivery point, but the secondary range is not available on input. *L*: The address triggered DPV locking. *N*: The address is not a valid delivery point. *S*: The primary range is a valid delivery point, but the parsed secondary range is not valid in the DPV directory. *Y*: The address is a confirmed delivery point. The primary range and secondary range (if present) are valid. **`<blank>`**: A blank output value indicates that Enable_DPV_Validation is set to No, DPV processing is currently locked, or the transform cannot assign the input address.
DPV_Vacant	Vacant address indicator. *Y*: The address is vacant. *N*: The address is not vacant. **`<blank>`**: The address was not looked up. ℹ Note The US Addressing report contains DPV Vacant counts in the DPV Summary section.
DSF2_Business_Indicator	Residential/business indicator. You may use this information to lower your parcel-shipping costs. (Some parcel delivery services charge more for delivery to residential addresses.) *Y*: Business address. *N*: Not a business address.

Field	Description
	`<blank>`: The address was not looked up.
DSF2_Delivery_Type	Delivery type. *1*: Curb-side delivery. *2*: NDCBU (Neighborhood Delivery Centralized Box Unit) delivery. *3*: Central delivery. *4*: Door-slot delivery. `<blank>`: The address was not looked up.
DSF2_Drop_Count	Drop count. If DSF2_Drop_Indicator contains Y or DPV_CMRA contains Y, then this field contains a value from 000 to 999, indicating the number of businesses or families served by this delivery point.
DSF2_Drop_Indicator	Drop indicator. *Y*: The delivery point serves multiple businesses or families. For example, delivery point may be a CMRA (Commercial Mail Receiving Agency). *N*: The delivery address is not a CMRA. `<blank>`: The address was not looked up.
DSF2_Educational_Ind	Educational indicator. *Y*: The address is an educational institution. *N*: The address is not an educational institution. `<blank>`: The address was not looked up.
DSF2_LACS_Conversion_Indicator	LACS (Locatable Address Conversion System) indicator. *Y*: The address is LACS convertible. *N*: The address is not LACS convertible. `<blank>`: The address was not looked up.
DSF2_Record_Type	Record type. *B*: Business address. *R*: Residential address. *U*: Unknown. AP.DSF_Deltype is blank. `<blank>`: No information available.
DSF2_Seasonal_Indicator	Seasonal address indicator.

Field	Description
	Y: The address is seasonally occupied. *N*: The address is not seasonal. **`<blank>`**: The address was not looked up.
DSF2_Throwback_Indicator	Throwback indicator. *Y*: Customer with street address wants delivery at PO Box instead. *N*: No throwback necessary. **`<blank>`**: The address was not looked up.
Error	Specifies the error status generated as the result of looking up the current record and performing a suggestion processing. Possible output values are 0–5. *0:* There are no suggestion selection errors. *1:* The necessary selection information is blank. For example, a lastline suggestion list is generated, but there is no lastline selection input field data to make a choice. *2:* The suggestion selection is invalid. For example, 8 was selected but there are only five suggestions. *3:* The suggestion entry in the input field is invalid. *4:* The suggestion range in the input field is invalid. *5:* The suggestion secondary range in the input field is invalid.
EWS_Match	Returns the results of the EWS (Early Warning System) match. *T*: True, the address is located in the EWS directory and is an EWS match. *F*: False, the address is not located in the EWS directory. **`<blank>`**: EWS is not enabled.
Extra1-10	Any non-address data found above or below the address data in the address block. Available only if the input data is presented through multiline fields.
Extraneous_Secondary_Address_Data	Consists of the data from Extraneous_Secondary_Unit_Number and Extraneous_Secondary_Non_Postal respectively. Any additional # data is placed in the remainder or extra components. This may include invalid secondary address data.
Extraneous_Secondary_Non_Postal	Extraneous data retained in this field is the best guess at Private Mail Box data, based on the position in the address line and other information contained in the address, such as a pound unit designator (#). This may include invalid secondary address data.

Field	Description
Extraneous_Secondary_Unit_Number	Extraneous data retained in this field is the best guess at secondary range data, based on the position in the address line and other information contained in the address. This may include invalid secondary address data.
Fault_Code	The fault code. This is blank if the address is assigned. For more information, see *Fault codes (USA Regulatory Address Cleanse)* [page 1417].
Fault_Or_Status_Code	The fault code if the address is unassigned; the status code if the address is assigned.
Finance_Area_Postcode	The Finance Area Postcode is the lowest Postcode1 within a Finance Number. (Finance Numbers are currently used to link data to a single post office or postmaster.)
Firm	Firm name. Do not use this field if the input was multiline, because if there is no firm name in the postal directory, the transform cannot reliably identify firm names from multilines. If you retrieve the corrected component, the firm name is taken from the postal directory if found; otherwise, it's taken from the input record. Be aware that the postal directory might contain some unusual or shortened spellings that you may or may not find suitable for printing on mail pieces. If you prefer to retain your own firm data, retrieve the original component.
Foreign_Code	Specifies whether the address is foreign or domestic. *F:* Foreign addresses **`<blank>`**: Domestic U.S.
Full_Address	The complete address line, including secondary address and dual address (street and postal) line data. i Note This field may contain invalid secondary address information when you set the *Include Unused Address Line Data* option in the Standardization Options group to *Yes*. i Note If the output values don't fit within the length of the output field, then intelligent truncation occurs.
Geo_Matchcode	Match code indicating the precision of the latitude and longitude assignment. *0:* Matches in address level.

Field	Description
	1: Nine-digit match. Usually indicates precision to a particular block face. *4:* Seven-digit match. Usually indicates precision within a few blocks. *5:* Five-digit match. Usually indicates precision to within a mile or two. *7:* No match in centroid-level. *8:* No match in address-level. *9:* No match in centroid-level or address-level. **<blank>**: No centroid-level or address-level directory lookup performed.
Intermediate_Codes	Intermediate codes provide information that the USPS requires when you perform NCOALink certification or audit testing.
LACSCode	LACS (Locatable Address Conversion System) indicator. *T:* The address needs 9-1-1 conversion (from box to street address) and should be submitted to a LACS vendor. *F:* The address does not need conversion. **<blank>**: The address was not assigned.
LACSLink_Indicator	Returns the conversion status of addresses processed by LACSLink. *Y:* The address was converted by LACSLink (the LACSLink_Return_Code value is A). *N:* The address was looked up with LACSLink but not converted. *F:* The address was a false-positive. *S:* A LACSLink conversion was made, but it was necessary to drop the secondary information. **<blank>**: No LACSLink lookup attempted.
LACSLink_Query	Returns the pre-conversion address, populated only when LACSLink is turned on and a LACSLink lookup was attempted. This address is in the standard Pub. 28 format. However, when an address has both a unit designator and secondary unit, the unit designator is replaced by the pound character (#). **<blank>**: No LACSLink lookup attempted.
LACSLink_Return_Code	Returns the match status for LACSLink processing. *A:* LACSLink record match. A converted address is provided in the address data fields. *00:* No match and no converted address. *09:* LACSLink matched an input address to an old address, which is a "high-rise default" address; no new address is provided.

Field	Description
	14: Found a LACSLink record, but couldn't convert the data to a deliverable address. *92:* LACSLink record matched after dropping the secondary number from input address. **`<blank>`**: No LACSLink lookup attempted.
Lastline	Locality, region, and postal code together on one line. Note If the output values don't fit within the length of the output field, then intelligent truncation occurs.
Locality1	*Canada and USA engines:* Locality preferred by the postal authority. *Other engines:* City, town, locality, or suburb. Note If the output values don't fit within the length of the output field, then intelligent truncation occurs.
Locality1_Alternate	Preserves the input Locality1 if it is not recognized by the postal authority as a valid Locality1 name for the input address line, and the original Locality1 was changed because of address assignment rules. Also outputs the default Locality 1 for the assigned finance area when there is no input Locality1, or the input Locality1 is not valid in the assigned finance area.
Locality1_LLIDX	Yields a city name (locality1 name) that is more geographically precise than Locality1_Official. LLIDX (last-line index) is a USPS number that ties a ZIP+4 record to a particular city, state, and ZIP. Note If the output values don't fit within the length of the output field, then intelligent truncation occurs.
Locality1_Name	The city, town, or suburb. Locality names that are marked as invalid for mailing by the USPS are always preserved, never converted, regardless of the values set for the *Preserve Place Name* and *Assign With Input Locality* options.

Field	Description
Locality1_Official	The standardized locality name. When the input city name is tagged by the USPS as invalid for mailing, this field always yields a converted city name, no matter how the *Preserve Place Name* option is set.
Locality1_Official_ABBR	The official USPS abbreviation of the city name, if one is available. This field is blank if the full city name is less than 13 characters or if the full name is longer, but the USPS has not provided an official abbreviation.
Locality2	*USA engine:* Urbanization (Puerto Rican addresses only). *Other engines:* Additional city, town, locality, or suburb information.
Locality2_Official	Urbanization name; produced only when the address is in Puerto Rico.
LOT	Line-of-travel number.
LOT_Order	Line-of-travel sortation. *A:* Ascending. *D:* Descending.
Matched_Addressline_Indicator	Match level indicator. *T:* The address line is matched to a ZIP+4 record. *F:* The address line is not matched to a ZIP+4 record.
Matched_Lastline_Indicator	Match level indicator. *T:* The last line is matched to a City/ZCF record. *F:* The last line is not matched to a City/ZCF record.
Move_Effective_Date	The date that the move is effective as indicated on the change of address card sent to the USPS in the format yyyymm. The yyyymm format is returned from the NCOALink directories and is required by the USPS for audit purposes. To use it in a function or post it to an output file, you'll probably have to convert the format to mm/dd/yyyy first. This field is also populated when an ANKLink match is made.
Move_Type	Type of move record. *B:* Business (matched by company name). *F:* Family (matched by last name). *I:* Individual (matched by first and last name). This field is also populated when an ANKLink match is made.
Multiline1-12	A line that may contain any data. The type of data in this line may vary from record to record.

Field	Description
	ℹ Note These fields may contain invalid secondary address information when you set the *Include Unused Address Line Data* option in the*Standardization Options* group to *Yes*. ℹ Note For address data, if the output values don't fit within the length of the output field, then intelligent truncation occurs.
Name	The name of a person associated with the address.
NCOALink_Hint_Byte	This field is used for audit testing.
NCOALink_Return_Code	This field shows NCOALink return codes. To populate this field, set the *List Processing Mode* to one of the three available options: *Change of Address*, *Statistics Only*, or *Return Codes Only*. A brief description of the return codes appears on the NCOALink Processing Summary report. To print more detailed return code descriptions on the report, enable the *Generate Return Code Descriptions* option in the NCOALink Report Options group. This field is also populated when an ANKLink match is made.
Stage_Test_Record	This field is for stage testing only. It applies to NCOALink, CASS, DSF2 Augment, DSF2 Sequence, and DSF2 Invoice self-certifications. The USA Regulatory Address Cleanse transform populates the values of this field automatically to match the format required for stage testing.
Non_CASS_Firm	The firm match that is made by using the input ZIP+4 for missing or invalid firm information.
Non_CASS_Secondary_Address	The secondary address match that is made using the input ZIP+4 for missing or invalid secondary address information. ℹ Note If the output values don't fit within the length of the output field, then intelligent truncation occurs.
Non_CASS_Unit	The unit designator match that is made using the input ZIP+4 for missing or invalid unit designator information.
Non_CASS_Unit_Number	The unit designator match that is made using the input ZIP+4 for missing or invalid unit designator information.

Field	Description
Non_Postal_Secondary_Address	The complete non-postal secondary address (for example, "PMB 10" or "# 10"). Non-postal means that the mail is delivered through a private mailbox company rather than the USPS.
Non_Postal_Unit	Non-postal unit designator (PMB or #). Non-postal means that the mail is delivered through a private mailbox company rather than the USPS.
Non_Postal_Unit_Number	Non-postal secondary range (PMB number only, does not include designator). Non-postal means that the mail is delivered through a private mailbox company rather than the USPS.
Parsed_Firm	If the change of address is made based on a firm (company) name, the firm name is posted in this field. This field is also populated when an ANKLink match is made.
Postal_Box_Number	Post office box number.
Postcode_Full	The complete ZIP10 with a hyphen.
Postcode_Full_No_Hyphen	The complete ZIP9 without a hyphen.
Postcode_Type	The type of ZIP Code that is assigned. *M:* Military. *U:* Unique (specific to a university, large firm, or other institution). **`<blank>`**: Ordinary ZIP Code or the ZIP Code was not assigned.
Postcode1	The five-digit ZIP Code. Does not include the four-digit ZIP4.
Postcode1_Change_Ind	Indicates whether the address is affected by postal code realignment. *T:* True, the transform corrected the postal code (and the locality, if applicable). *F:* False. **`<blank>`**: The address was not corrected.
Postcode2	Four-digit ZIP4 Code. On a mail piece, this code follows the primary postal code, with a hyphen placed between, for example, 54601-1234.
Pre_Suitelink_Delivery_Point	The numeric two-digit code for the delivery point bar code that was generated before SuiteLink processing.
Pre_Suitelink_Postcode1	The ZIP Code that was assigned by the transform before SuiteLink processing. *5-digit ZIP Code*: SuiteLink Retcode value is A. **`<blank>`**: No ZIP Code assigned.

Field	Description
Pre_Suitelink_Postcode2	The ZIP+4 that was assigned by the transform before SuiteLink processing. The ZIP+4 is either for a high-rise default or street default record.
Pre_Suitelink_Unit_Description	The unit designator that existed before SuiteLink processing. If this field is blank, the transform did not assign any secondary information.
Pre_Suitelink_Unit_Number	The secondary range information that existed before SuiteLink processing. If this field is blank, the transform did not assign any secondary information.
Primary_Address	Primary address line, such as the street address or post office box. Does not include secondary address information such as apartment. If the *Use USPS Primary Name Abbreviation* option is enabled, the software uses the USPS Primary Name abbreviation first. If the values don't fit within the length of the output fields, then intelligent truncation occurs.
Primary_Name1	Street name description. Note If the output values don't fit within the length of the output field, then intelligent truncation occurs.
Primary_Number	The premise number.
Primary_Postfix1	An abbreviated directional (such as N, S, NW, or SE) that follows a street name.
Primary_Postfix1_Long	A fully-spelled directional (such as North or South) that follows the street name.
Primary_Prefix1	An abbreviated directional (such as N, S, NW, or SE) that precedes a street name.
Primary_Prefix1_Long	A fully-spelled directional (such as North or South) that precedes the street name.
Primary_Secondary_Address	The primary address and secondary address on one line. Does not include remainder data. This line is always output as if the *Include Unused Address Line Data* option is set to *No*, which means that the output does not include invalid secondary address line information. Note If the output values don't fit within the length of the output field, then intelligent truncation occurs.
Primary_Type1	Abbreviated street type (for example, St, Ave, or Pl).
Primary_Type1_Long	Fully-spelled street type (for example, Street or Avenue).

Field	Description
QSS_Default	Specifies whether the record qualified as a default match instead of qualifying as a match at a higher level of assignment. Output values are: *T:* True *F:* False
RDI_Indicator	The residential delivery indicator (RDI) shows whether the address is residential or nonresidential. *Y:* Residential address *N:* Nonresidential address
Region1	State, province, territory, or region.
Rural_Route_Box_Number	The rural route box number.
Rural_Route_Number	The rural route number.
Secondary_Address	The building name, floor, and room number in one field.
Sortcode_Postcode	The federal code for state and county (FIPSCODE). Combines the two-digit state code with the three-digit county code. U.S. territories, possessions, or protectorates such as Puerto Rico, the U.S. Virgin Islands, or the Pacific Islands do not have FIPS state digits.
Sortcode_Route	The four-digit carrier route.
Status	Specifies the suggestion status generated as the result of looking up the current record and performing suggestion processing. *A:* Suggestion processing ended with an address suggestion list needing resolution. *L:* Suggestion processing ended with a lastline suggestion list needing resolution. *N:* No suggestion lists were generated and no suggestion processing was performed. *R:* The primary range is invalid for the selected address suggestion. *S:* The secondary range is invalid for the selected address suggestion. *U:* The secondary address is invalid for the selected address suggestion.
Status_Code	The status code. This field is blank if the address is unassigned. For more information, see *Status codes (USA Regulatory Address Cleanse)* [page 1418].
Suggestion_List	Contains all of the Suggestion List Component field values that you chose in the Suggestion List group of the USA Regulatory Address Cleanse transform.

Field	Description
SuiteLink_Retcode	*A:* SuiteLink match—Secondary information exists and was assigned to this record as a result of SuiteLink processing. *00:* No SuiteLink match—Lookup was attempted but no matching record was found. **<blank>**: A SuiteLink lookup was not attempted because one of the following is true: • The address is not a high-rise default according to CASS. • The address does not contain a firm.
Undeliverable_Indicator	Indicates whether the record is a deliverable address. *T:* The address is tagged by the USPS as unsuitable for mail delivery (for example, a cemetery). *F:* The address either was not matched to a ZIP+4 record or was matched to a record that indicates that the address is suitable for mail delivery.
Unit_Description	Unit description (for example, #, Apartment, or Flat).
Unit_Description_Directory	Unit designator from ZIP+4 directory, or blank if none was found.
Unit_Number	Unit number (for example, 100 in "APT 100").

Related Information

Fault codes (USA Regulatory Address Cleanse) [page 1417]
Status codes (USA Regulatory Address Cleanse) [page 1418]

3.5.4.14 User-Defined

The User-Defined transform provides you with custom processing in a data flow using full Python scripting language. The applications for the User-Defined transform are nearly limitless. It can do just about anything that you can write Python code to do. You can use the User-Defined transform to generate new records, populate a field with a specific value, create a file, connect to a website, or send an email, just to name a few possibilities.

You can place this transform anywhere in your data flow. If you have created your own transform, then the only restrictions about where it can be located in the data flow are those which you place on it.

Although the User-Defined transform is quite flexible and powerful, you will find that many of the tasks you want to perform can be accomplished with the Query transform. The Query transform is generally more scalable and faster, and uses less memory than User-Defined transforms.

Editors

Like all Data Quality transforms, the User-Defined transform has a transform editor which contains the Input, Options, and Output tabs.

Unlike most of the other Data Quality transforms, you cannot edit options in the User-Defined transform editor. To edit options, you must use the User-Defined editor, which is accessed from the Options tab in the User-Defined transform editor or from the Tools menu.

You may also notice some options displayed in the Options tab of the User-Defined transform editor that are not displayed in the User-Defined editor. These options are not editable.

⚠ Caution

Make sure that if you use an input field in a Python expression in your User-Defined transform, you first map it to a recognized field name in the Input tab. If it is not mapped, you will receive an error message similar to the following:

def GetField(*args): return apply(_flpythonmodulesu.FlDataRecord_GetField,args) RuntimeError: FlDataRecord::GetField() error: Invalid field name MAPPED_RECNO.

Related Information

Designer Guide: Data Flows, Associate, Match, and User-Defined transform editors [page 316]
Designer Guide: Data Flows, Data Quality transform editors [page 314]

3.5.4.14.1 Content objects

Transform configurations

A transform configuration is a transform with preconfigured input fields, output fields, and options that can be used in multiple data flows. These are useful if you repeatedly use a transform with specific options and input and output fields.

When Data Services is installed, read-only transform configurations are provided for the Data Quality transforms.

You can use transform configurations in your data flows or as an example of a typical transform. After you place an instance of the transform configuration in a data flow, you can override these preset defaults. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one.

Sample blueprints and other objects

We have created Data Quality blueprints and other content objects to help you set up Data Services jobs. We've identified a number of common scenarios that you are likely to perform with Data Services. For each scenario, we've included a blueprint that is already set up to solve the business problem in that scenario.

Related Information

Downloading blueprints and other content objects [page 1128]
Transform configurations [page 1123]

3.5.4.14.2 User-Defined options

The User-Defined transform contains options that determine how the transform processes data. Many User-Defined transform options are also found in the Match transform.

Option	Description
Mode	Specifies how the Python expression is applied to the transform. *Per collection:* Applies the expression to entire data collection. Use this option when adding new records, which did not exist before, into the data flow. Selecting this option displays the Group Forming option group, in which you set up break groups and candidate selection. *Per record:* Applies the expression to each record. You cannot add new records into the data flow with this option. This option is the default and what you will want to use most often.

Related Information

Group forming [page 1373]
User-Defined Transform options [page 1378]

3.5.4.14.2.1 Group forming

Group forming allows you to group and prioritize records for better match accuracy and efficiency, as well as performing custom Python processing.

Break groups

Break groups allow you to group records based on common field values.

Use break groups to lower the number of comparison the Match transform needs to make and to increase the speed of the matching process.

Candidate selection

The process of candidate selection appends records from a relational database to an existing data collection for processing.

For real-time jobs, candidate selection pulls a candidate set of records based on a single record or many records.

To display the Candidate Selection option group, right-click the Group Forming option group and select Add Candidate Selection.

> ℹ Note
>
> Candidate selection works with relational databases only; it does not work with flat files.

Related Information

Break group options [page 1270]
Candidate selection options [page 1272]

3.5.4.14.2.1.1 Break group options

Use the break group options to group records based on common field values.

Options	Description
Split records into break groups	Select this option if you want to form break groups to reduce the total number of comparisons made. The most common case for deselecting this option is when have a real-time job and your data comes in as one break group. This scenario also often makes use of candidate selection (selecting a limited number of records from a relational database) for optimal real-time matching. ⚠ Caution Deselect this option with caution within a batch data flow. The size of a break group may not exceed 2 GB. If you use this option in a batch data flow, also

Options	Description
	set the *Maximum allowable break group size (in records)* option so that the collection does not exceed the size limit. If it does exceed the limit, the data flow will abort. Note Break group size is calculated by multiplying the record length by the number of records in the break group.
Field	Choose a mapped input field name from the drop-down menu that you want to include in the break key. Click the Add Row button to add another field. If you require a more complex break key, you could define that field using an upstream Query transform and select the field here.
Start Position	Enter the start position of the field. Valid values for a field of n are 1 to n and -1 to -n. Negative start values signify that the start position is counted from the right. For example, a field with a length of 7 contains JOHNSON. A start position of 2 would mean start with "O." A start position of -4 means start with the "N" (This would also be the case if the field has a length of 20, because the negative start value starts from the actual length of the string, not of the field).
Length	Enter the number of characters in the field you want included in the break key.
Break key case sensitive	Specifies whether to treat the break key as case sensitive. *Yes:* Treat the break key as case sensitive. *No:* Do not treat the break key as case sensitive. For example, if you create a break key using the primary name (street), separate break groups would be formed with values of “Main” and “main” when you specify that the break key is case sensitive.
Replace NULL with empty string	Specifies whether to convert NULL values with an empty string in the break key. *Yes:* Convert NULL to an empty string. *No:* Do not convert to an empty string.
Right pad fields with blanks	Because the break key is used for sorting and aggregating, it is sensitive to the position in which data is placed. By right-padding the break key fields you can help ensure that break groups are formed properly. If the *Replace NULL with empty string* option is set to YES and this option is set to YES, then fields with NULL values will be replaced with all spaces (to the length of the field). *Yes:* Right-pad fields with blank spaces. *No:* Do not right-pad fields.

Options	Description
Input already sorted	Specifies that the input data has already been sorted, and you do not want it sorted again. For example, if you require a more complex break key, you could use a Query transform to create it, and use the ORDER BY operation to order your data. *Yes:* The transform will not re-sort the input data. *No:* The transform will sort the break keys at runtime before forming break groups.
Maximum allowable break group size (in records)	Specifies the maximum number of records allowed in a break group. An empty value or zero means that there is no limit on the break group size. With this option, you can control the amount of memory used during processing by specifying the number of records processed at one time. If more records make it into a single break group than specified, then the data flow throws an error and stops.

Related Information

Designer Guide: Match, Break keys and candidate selection [page 533]

3.5.4.14.2.1.2 Candidate selection options

The candidate selection option group includes the following options:

Option	Description
Datastore	Select a valid datastore. This list is populated with all valid SQL and persistent cache datastores. If you choose a persistent cache datastore, you will not be able to enter custom SQL.
Cache type	This option can be used to improve performance, with a trade-off of more memory consumption. *No_Cache:* Specifies that each query will be sent to the database. *Pre_Load_Cache:* Specifies that the entire secondary table is cached to a local disk or memory.
Auto-generate SQL	Select to have your SQL generated by the transform. This option allows you to query a simple single table. If you need to join tables or create a complex WHERE clause, you should select the *Create custom SQL* option.

Option	Description
Table	Enter a valid table name from the datastore.
Use break column from database	Select this option if your database already contains a column that corresponds to the break key field.
Break key field	Select the column from the secondary table that contains the break key field.
Create custom SQL	Select to create custom SQL.
Launch SQL Editor	Opens the SQL editor. This button is only enabled if you select the *Create custom SQL* option.
Use constant source value	Select to assign records to a physical source for generating appropriate statistics.
Physical source value	Type a value for your physical source. This value will be placed in the physical source field you select.
Physical source field	Select the mapped field that contains the physical source name.
Add DB columns to mapping table	If you are using the *Create custom SQL* option, clicking this button will add only the database columns that appear in the SELECT statement and in the order that they appear in the SELECT statement. If you are using the *Auto-generate SQL* option, clicking this button will add ALL database columns, in the order that they appear in the table schema. Note If you do not associate an input field to any of these columns in the column mapping table, they will be removed when you close the window.

Column mapping table

This table allows you to specify which mapped field in the data flow each database selected field is assigned to.

Column	Description
Break key	Specifies whether this field is used as part of your break key.
Field	Each cell contains a list of the mapped names from the input fields in the transform.
DB column	Each cell contains a list of the column names in your database table or the selected columns from a custom query. Match the data of a column in your database to the data of a mapped field.

Related Information

Designer Guide: Match, Break keys and candidate selection [page 533]

3.5.4.14.2.2 User-Defined Transform options

Custom options table

This table allows you to create custom options to be used as variables in your Python expression. The custom options are only available within the User-Defined transform. These options adjust the User-Defined transform's run-time behavior. Add or remove rows by using the buttons.

Option	Description
Custom option	Specifies variables for use in your Python expression.
Value	Specifies the value of the custom option.

The User-Defined Transform option group also contains the following option:

Option	Description
Run as separate process	This option creates a separate sub data flow process for the transform when Data Services executes the data flow. *Yes:* Splits transform into separate process. *No:* Keeps transform in same process as the rest of the data flow.

3.5.4.14.2.2.1 Python Expression Editor option

This option group contains the actual Python expression that the User-Defined transform will use. This option group is required.

Click the Launch Python Editor button to access the Python Expression editor.

Option	Description
Python	Displays the Python expression that will be applied to the transform. You can enter the Python code here or use the Python Expression editor.

⚠ Caution

Make sure to use a "u" to indicate Unicode every time you use a Unicode string to look up field names; for example, in a GetField, SetField, or SendToPipe method. If you do not, an error or crash may occur.

Related Information

Python [page 1813]
Create an expression with the Python Expression editor [page 1817]

3.5.4.15 Address Cleanse reference

This section describes reference information for use with the Address Cleanse transforms (Global Address Cleanse and USA Regulatory Address Cleanse). For the USA Regulatory Address Cleanse transform, it lists the available certifications and the steps to achieve those certifications.

This section also explains how to use the Show A and Show L utilities (for United States addresses only) that you can use to query the postal directories used by either of the Address Cleanse transforms.

For the Global Address Cleanse transform:

- Information codes
- Status codes
- Quality codes

For the USA Regulatory Address Cleanse transform:

- Status codes
- Fault codes

3.5.4.15.1 Beyond the basic address cleansing

The USA Regulatory Address Cleanse transform offers many additional address cleanse features for U.S. addresses. These features extend address cleansing beyond the basic parsing and standardizing. To read about the USA Regulatory Address Cleanse transform and its options, see the *Reference Guide*.

3.5.4.15.1.1 USPS certifications

The USA Regulatory Address Cleanse transform is CASS-certified. Therefore, when you process jobs with the USA Regulatory Address Cleanse transform (and it is set up correctly) you reap the benefits of that certification.

If you integrate Data Services into your own software and you want to obtain CASS certification, you must follow the steps for CASS self-certification using your own software.

You can also obtain licenses for DSF2 (Augment, Invoice, Sequence) and for NCOALink by using USA Regulatory Address Cleanse and DSF2 Walk Sequencer blueprints that are specifically set up for that purpose.

Note

In this section we direct you to the USPS website and include names of documents and procedures. The USPS may change the address, procedure, or names of documents (and information required) without our prior knowledge. Therefore some of the information may become outdated.

Related Information

CASS self-certification [page 1383]
DSF2 Certification [page 1392]
Designer Guide: Getting started with NCOALink [page 656]

3.5.4.15.1.1.1 Completing USPS certifications

The instructions below apply to USPS CASS self-certification, DSF2 license, and NCOALink license certification.

During certification you must process files from the USPS to prove that your software is compliant with the requirements of your license agreement.

The CASS, DSF2, and NCOALink certifications have two stages. Stage I is an optional test which includes answers that allow you to troubleshoot and prepare for the Stage II test. The Stage II test does not contain answers and is sent to the USPS for evaluation of the accuracy of your software configuration.

1. Complete the applicable USPS application (CASS, DSF2, NCOALink) and other required forms and return the information to the USPS.
   After you satisfy the initial application and other requirements, the USPS gives you an authorization code to purchase the CASS, DSF2, or NCOALink option.
2. Purchase the option from the USPS. Then submit the following information to SAP:
   - your USPS authorization code (see step 1)
   - your NCOALink provider level (full service provider, limited service provider, or end user) (applicable for NCOALink only)
   - your decision whether or not you want to purchase the ANKLink option (for NCOALink limited service provider or end user only)

   After you request and install the feature from SAP, you are ready to request the applicable certification test from the USPS. The software provides blueprints to help you set up and run the certification tests. Import them from **`<DS_COMMON_DIR>`**`\DataQuality\Certifications`, where **`<DS_COMMON_DIR>`** is the software's common configuration directory.
3. Submit the Software Product Information form to the USPS and request a certification test.
   The USPS sends you test files to use with the blueprint.
4. After you successfully complete the certification tests, the USPS sends you the applicable license agreement. At this point, you also purchase the applicable product from SAP.

Related Information

Designer Guide: About ANKLink [page 655]
To set up the NCOALink blueprints [page 1390]
To set up the DSF2 certification blueprints [page 1394]

3.5.4.15.1.1.2 Introduction to static directories

Users who are self-certifying for CASS must use static directories. Those obtaining DSF2 licenses also need to use static directories. Static directories do not change every month with the regular directory updates. Instead, they can be used for certification for the duration of the CASS cycle. Using static directories ensures consistent results between Stage I and Stage II tests, and allows you to use the same directory information if you are required to re-test. You obtain static directories from SAP.

> Note
>
> If you do not use static directories when required, the software issues a warning.

3.5.4.15.1.1.2.1 Static directories

The following directories are available in static format:

- `zip4us.dir`
- `zip4us.shs`
- `zip4us.rev`
- `revzip4.dir`
- `city10.dir`
- `zcf10.dir`
- `dpv*.dir`
- `elot.dir`
- `ew*.dir`
- SuiteLink directories
- LACSLink directories

3.5.4.15.1.1.2.2 Obtaining static directories

To request static directories, contact SAP Business User Support. Contact information (toll-free number and email address) is available at *http://service.sap.com*.

1. Click *SAP Support Portal*.

2. Click the *Help and Support* tab.
3. Click *Contact SAP*.
4. Complete the form and then click *Submit*.

3.5.4.15.1.1.2.3 Static directories location

It is important that you store your static directories separately from the production directories. If you store them in the same folder, the static directories will overwrite your production directories.

We suggest that you create a folder named “static” to store your static directories. For example, save your static directories under **<DS_COMMON_DIR>**`\DataQuality\reference\static`, where **<DS_COMMON_DIR>** is the software's common configuration directory.

3.5.4.15.1.1.2.4 Static directories safeguards

To prevent running a production job using static directories, the software issues a verification warning or error under the following circumstances:

- When the job has both static and non-static directories indicated.
- When the release version of the `zip4us.dir` does not match the current CASS cycle in the software.
- When the data versions in the static directories aren't all the same. For example, for CASS Cycle M the data versions in the static directories are M01.
- When the job is set for self-certification but is not set up to use the static directories.
- When the job is not set for self-certification but is set up to use the static directories.

3.5.4.15.1.1.3 To import certification blueprints

The software includes blueprints to help you with certification testing. Additionally, the blueprints can be used to process a test file provided by the USPS during an audit. You need to first import the blueprints to the repository before you can use them in Data Services.

To import the certification blueprints, follow these steps:

1. Open Data Services Designer.
2. Right-click in the Object Library area and select *Repository* > *Import from file*.
3. Go to **<DS_COMMON_DIR>**`\DataQuality\certifications`, where **<DS_COMMON_DIR>** is the software's common configuration directory.
4. Select the applicable blueprint and click *Open*.

> i Note
>
> A message appears asking for a pass phrase. The blueprints are not pass phrase protected, just click *Import* to advance to the next screen.

5. Click *OK* at the message warning that you are about to import the blueprints.

   Importing the blueprint files into the repository adds new projects, jobs, data flows, and flat file formats. The naming convention of the objects includes the certification type to indicate the associated certification test.

Related Information

CASS self-certification blueprint [page 1384]
NCOALink blueprints [page 1389]
DSF2 Certification blueprints [page 1393]

3.5.4.15.1.1.4 CASS self-certification

If you integrate Data Services into your own software, and you want to CASS-certify your software, you must obtain CASS certification on your own (self certification). You need to show the USPS that your software meets the CASS standards for accuracy of postal coding and address correction. You further need to show that your software can produce a facsimile of the USPS Form 3553 . You need a USPS Form 3553 to qualify mailings for postage discounts.

Obtaining CASS certification on your own software is completely optional. However there are many benefits when your software is CASS certified.

Visit the USPS RIBBS website at *http://ribbs.usps.gov/index.cfm?page=cassmass* for more information about CASS certification.

Related Information

Completing USPS certifications [page 1380]

3.5.4.15.1.1.4.1 CASS self-certification process overview

1. Familiarize yourself with the CASS certification documentation and procedures located at *http://ribbs.usps.gov/index.cfm?page=cassmass* .
2. (Optional.) Download the CASS Stage I test from the RIBBS website.
   This is an optional step. You do not submit the Stage I test results to the USPS. Taking the Stage I test helps you analyze and correct any inconsistencies with the USPS-expected results before taking the Stage II test.
3. Import and make modifications to the CASS self-certification blueprint (`us_cass_self_certification.atl`). The blueprint is located in **`<DS_COMMON_DIR>`**`\DataQuality\Certifications`, where **`<DS_COMMON_DIR>`** is the software's common configuration location.
   Edit the blueprint so it contains your static directories location, Stage I file location, your company name, and other settings that are required for CASS processing.

4. When you are satisfied that your Stage I test results compare favorably with the USPS-expected results, request the Stage II test from the USPS using the Stage II order form located on the RIBBS website.
   The USPS will place the Stage II test in your user area on the RIBBS website for you to download.
5. Download and unzip the Stage II test file to an output area.
6. After you run the Stage II file with the CASS self-certification blueprint, check that the totals on the USPS Form 3553 and the actual totals from the processed file match.
7. Compress the processed Stage II answer file (using WinZip for example) and name it using the same name as the downloaded Stage II file (step 5).
8. Upload the processed Stage II answer file to your user area on the RIBBS website.

The USPS takes about two weeks to grade your test.

3.5.4.15.1.1.4.2 CASS self-certification blueprint

SAP provides a CASS self-certification blueprint. The blueprint contains the corresponding project, job, data flow, and input/output formats. Additionally, the blueprint can be used to process a test file provided by the USPS during an audit.

Import the `us_cass_self_certification.atl` blueprint from **`<DS_COMMON_DIR>`**`\DataQuality\Certifications` where **`<DS_COMMON_DIR>`** is the software's common configuration location. The table below contains the file names for the CASS self-certification blueprint:

Object	Name
ATL file	us_cass_self_certification.atl
Project	DataQualityCertificationCASS
Job	Job_DqBatchUSAReg_CASSSelfCert
data flow	DF_DqBatchUSAReg_CASSSelfCert
Input file format	DqUsaCASSSelfCert_In
Output file format	DqUsaCASSSelfCert _Out

3.5.4.15.1.1.4.3 USPS Form 3553 required options for self certification

The following options in the CASS Report Options group are required for CASS self certification. This information is included in the USPS Form 3553.

Option	Description
Company Name Certified	Specify the name of the company that owns the CASS-certified software.
List Name	Specify the name of the mailing list.

Option	Description
List Owner	Specify the name of the list owner. ℹ Note Keep the CASS self-certification blueprints setting of "USPS".
Mailer Address(1-4)	Specify the name and address of the person or organization for whom you are preparing the mailing (up to 29 characters per line).
Software Version	Specify the software name and version number that you are using to receive CASS self certification.

3.5.4.15.1.1.4.4 Points to remember about CASS

Remember these important points about CASS certification:

- As an end user (you use Data Services to process your lists), you are not required to obtain CASS self certification because Data Services is already CASS certified.
- CASS certification is given to software programs. You obtain CASS self certification if you have incorporated Data Services into your software program.
- The CASS reports pertain to address lists.
- CASS certification proves that the software can assign and standardize addresses correctly.

3.5.4.15.1.1.5 NCOALink certification

NCOALink is a feature that must be purchased before testing.

The NCOALink certification consists of the following steps:

1. Application and Self-Certification Statement Approval
2. Software acquisition
3. Testing and certification
4. Execution of License Agreement

This entire procedure is documented in the USPS Certification Procedures documents posted on the RIBBS website at *http://ribbs.usps.gov/ncoalink/documents/tech_guides* . Select either NCOALink End User Documents, NCOALink Limited Service Provider Documents, or NCOALink Full Service Provider Documents as applicable.

You must complete the appropriate service provider certification procedure for NCOALink in order to purchase the NCOALink product from the USPS.

Related Information

Designer Guide: Getting started with NCOALink [page 656]

3.5.4.15.1.1.5.1 About NCOALink directories

After you have completed the certification requirements and purchased the NCOALink product from the USPS, the USPS sends you the latest NCOALink directories monthly (if you're an end user) or weekly (if you're a limited or full service provider). The NCOALink directories are not provided by SAP.

The USPS requires that you use the most recent NCOALink directories available for your NCOALink jobs.

> i Note
>
> The NCOALink directories expire within 45 days.

The software provides a DVD Verification (Installer) utility that installs (transfers and unpacks) the compressed files from the NCOALink DVD onto your system. The utility is available with a GUI (graphical user interface) or you can run it from a command line.

If you are a service provider, then each day you run an NCOALink job, you must also download the daily delete file and install it in the same folder where your NCOALink directories are located.

Related Information

Designer Guide: About the NCOALink daily delete file [page 659]
To install NCOALink directories with the GUI [page 1386]

3.5.4.15.1.1.5.1.1 To install NCOALink directories with the GUI

Prerequisites

Ensure your system meets the following minimum requirements:

- At least 60 GB of available disk space
- DVD drive
- Sufficient RAM.

1. Insert the USPS DVD containing the NCOALink directories into your DVD drive.
2. Run the DVD Installer, located at `$LINK_DIR\bin\ncoadvdver.exe` (Windows) or `$LINK_DIR/bin/ncoadvdver` (UNIX), where `$LINK_DIR` is the path to your software installation directory.

   For further installation details, see the online help available within the DVD Installer (choose *Help* > *Contents*).

For more information about required disk space for reference data, see the Product Availability Matrix at *https://service.sap.com/PAM*.

Related Information

SAP Business Objects information resources, Supported platforms (Product Availability Matrix) [page 169]

3.5.4.15.1.1.5.1.2 To install NCOALink directories from the command line

Prerequisites:

Ensure your system meets the following minimum requirements:

- At least 60 GB of available disk space
- DVD drive
- Sufficient RAM

1. Run the DVD Installer, located at `$LINK_DIR\bin\ncoadvdver.exe` (Windows) or `$LINK_DIR/bin/ncoadvdver` (UNIX), where `$LINK_DIR` is the path to your installation directory.
2. To automate the installation process, use the `ncoadvdver` command with the following command line options:

Option		Description
Windows	UNIX	
	-c	Run selected processes in console mode (do not use the GUI).
/p:t	-p:t	Perform transfer. When using this option you must also specify the following: ○ DVD location with /d or -d ○ transfer location with /t or -t
/p:u	-p:u	Perform unpack. When using this option, you must also specify the following: ○ DVD location with /d or -d ○ transfer location with /t or -t
/p:v	-p:v	Perform verification. When using this option, you must also specify the transfer location with /t or -t.
/d	-d	Specify DVD location.
/t	-t	Specify transfer location.
/nos	-nos	Do not stop on error (return failure code as exit status).

Option		Description
Windows	UNIX	
/a	-a	Answer all warning messages with Yes.

You can combine p options. For example, if you want to transfer, unpack, and verify all in the same process, enter /p:tuv or -p:tuv.

After performing the p option specified, the program closes.

Example

Your command line may look something like this:

Windows

```
ncoadvdver /p:tuv /d D:\ /t C:\pw\dirs\ncoa
```

UNIX

```
ncoadvdver [-c] [-a] [-nos] [-p:(t|u|v)] [-d<path>] [-t <filename>]
```

3.5.4.15.1.1.5.2 NCOALink software product information

Use the information below to complete the Compliance Testing Product Information Form. Find this form on the RIBBS website at http://ribbs.usps.gov/ncoalink/documents/tech_guides . Click the Compliance Testing Form.doc link.

Compliance Testing Product Information form	Description
Company Name & License Number	Your specific information. The license number is the authorization code provided in your USPS approval letter.
Company's NCOALink Product Name	Mover ID for NCOALink
Platform or Operating System	Your specific information
NCOALink Software Vendor	SAP Americas, Inc.
NCOALink Software Product Name	Mover ID
NCOALink Software Product Version	ACE
Address Matching ZIP+4 Product Name	Contact SAP Business User Support.
Address Matching ZIP+4 Product Version	Contact SAP Business User Support.
Address Matching ZIP+4 System	Closed
Is Software Hardware Dependent?	No
DPV® Product Name	ACE
DPV Product Version	Contact SAP Business User Support.

Compliance Testing Product Information form	Description
LACSLink® Product Name	ACE
LACSLink Product Version	Contact SAP Business User Support.
NCOALink Software options: Integrated or Standalone check boxes	Integrated
ANKLink Enhancement check box (applicable for Limited Service Providers and End Users)	Check the box if you purchased the ANKLink option from SAP.
HASH—FLAT—BOTH check boxes	Indicate your preference. The software provides access to both file formats.
NCOALink Level Option check boxes	Check the appropriate box.

Related Information

Completing NCOALink certification [page 1389]

Designer Guide: Data format [page 662]

3.5.4.15.1.1.5.3 Completing NCOALink certification

During certification you must process files from the USPS to prove that you adhere to the requirements of your license agreement. NCOALink certification has two stages. Stage I is an optional test which includes answers that allow you to troubleshoot and prepare for the Stage II test. The Stage II test does not contain answers and is sent to the USPS for evaluation of the accuracy of your software configuration.

Related Information

To run the NCOALink certification jobs [page 1391]

3.5.4.15.1.1.5.4 NCOALink blueprints

SAP provides NCOALink blueprints. The blueprints contain the corresponding projects, jobs, data flows, and input/output formats. Additionally, the blueprints can be used to process a test file provided by the USPS during an audit.

Import NCOALink blueprints from **`<DS_COMMON_DIR>`**`\DataQuality\Certifications` where **`<DS_COMMON_DIR>`** is the software's common configuration location.

The table below contains the file names for the Stage I NCOALink blueprints:

Object	Name
ATL file	us_ncoalink_stage_certification.atl
Project	DataQualityCertificationNCOALink
Job	Job_DqBatchUSAReg_NCOALinkStageI
data flow	DF_DqBatchUSAReg_NCOALinkStageI
Input file format	DqUsaNCOALinkStageI _in
Output file format	DqUsaNCOALinkStageI _out

The table below contains the file names for the Stage II NCOALink blueprints:

Object	Name
ATL file	us_ncoalink_stage_certification.atl
Project	DataQualityCertificationNCOALink
Job	Job_DqBatchUSAReg_NCOALinkStageII
data flow	DF_DqBatchUSAReg_NCOALinkStageII
Input file format	DqUsaNCOALinkStageII _in
Output file format	DqUsaNCOALinkStageII _out

3.5.4.15.1.1.5.5 To set up the NCOALink blueprints

Before performing the steps below you must import the NCOALink blueprints.

To set up the NCOALink Stage I and Stage II blueprints, follow the steps below.

1. In the Designer, select ▶ *Tools* ▶ *Substitution Parameter Configurations* ◀.
   The *Substitution Parameter Editor* opens.
2. Choose the applicable configuration from the *Default Configuration* drop list and enter values for your company's information and reference file locations. Click *OK* to close the Substitution Parameter Configurations tool.
3. Open the DataQualityCertificationsNCOALink project (which was imported with the blueprints).
4. Open the Job_DqBatchUSAReg_NCOALinkStageI job and then open the DF_DqBatchUSAReg_NCOALinkStageI data flow.
5. Click the DqUsaNCOALinkStageI_in file to open the *Source File Editor*. Find the *Data Files(s)* property group in the lower portion of the editor and make the following changes:
   a) In the *Root Directory* option, type the path or browse to the directory containing the input file.
      If you type the path, do not type a backslash (\) or forward slash (/) at the end of the file path.
   b) In the *File name(s)* option, change StageI.in to the name of the Stage file provided by the USPS.
   c) Exit the *Source File Editor*.
6. Click the DqUsaNCOALinkStageI_out file to open the *Target File Editor*. In the *Data Files(s)* property group make the following changes:

   a) In the *Root Directory* option, type the path or browse to the directory containing the output file.

      If you type the path, do not type a backslash (\) or forward slash (/) at the end of the file path.
   b) (Optional.) In the *File name(s)* option, change `StageI.out` to conform to your company's file naming convention.
   c) Exit the *Target File Editor*.
7. Double-click the USARegulatoryNCOALink_AddressCleanse transform to open the Transform Editor and click the *Options* tab.
8. Enter the correct path location to the reference files in the Reference Files group as necessary. Use the $ $RefFilesAddressCleanse substitution variable to save time.
9. In the USPS License Information group, do the following:
   a) Enter a meaningful number in the *List ID* option.
   b) Enter the current date in the *List Received Date* and *List Return Date* options.
   c) Ensure that the provider level specified in the substitution parameter configuration by the $ $USPSProviderLevel is accurate or specify the appropriate level (Full Service Provider, Limited Service Provider, or End User) in the *Provider Level* option.
   d) If you are a full service provider or limited service provider, complete the options in the *NCOALink* > *PAF Details* group and the *NCOALink* > *Service Provider Options* group.
10. When you are satisfied with the results of the Stage I test, repeat steps 4 through 9 to set up the Stage II objects found in the DF_DqBatchUSAReg_NCOALinkStage II data flow.

Related Information

Reference Guide: USPS license information options [page 1323]

To import certification blueprints [page 1382]

DSF2 Certification blueprints [page 1393]

NCOALink blueprints [page 1389]

CASS self-certification blueprint [page 1384]

3.5.4.15.1.1.5.6 To run the NCOALink certification jobs

Before you run the NCOALink certification jobs, make sure that you have installed the DPV, LACSLink, and U.S. National directory files to the locations you specified during configuration and that the NCOALink DVD provided by the USPS is available.

Running the Stage I job is optional; the results do not need to be sent to the USPS. However, running the Stage I job can help you ensure that you have configured the software correctly and are prepared to execute the Stage II job.

1. Use the NCOALink DVD Verification utility to install the NCOALink directories provided by the USPS. (See the link below for information about the NCOALink DVD Verification utility.)
2. Download the current version of the USPS daily delete file from *https://epf.usps.gov/* .
3. Download the Stage I file from *http://ribbs.usps.gov/* and uncompress it to the location you specified when setting up the certification job.

Ensure the input file name in the source transform matches the name of the Stage I file from the USPS.

4. Execute the Stage I job and compare the test data with the expected results provided by the USPS in the Stage I input file.

   As necessary, make modifications to your configuration until you are satisfied with the results of your Stage I test.
5. Download the Stage II file from the location specified by the USPS and uncompress it to the location you specified when setting up the certification job.

   Ensure the input file name in the transform matches the name of the Stage II file from the USPS.
6. Execute the Stage II job. Follow the specific instructions in the *NCOALink Certification/Audit Instructions* document that the USPS should have provided to you.
7. Compress the following results (using WinZip for example) and name it using the same name as the downloaded Stage II file (step 5):
   - Stage II output file
   - NCOALink Processing Summary Report
   - CASS Form 3553
   - All log files generated in the `$$CertificationLog` path
     - Customer Service Log
     - PAF (Service Providers only)
     - Broker/Agent/List Administrator log (Service Providers only)
8. Send the results to the USPS for verification.

Related Information

Management Console Guide: Exporting NCOALink certification logs [page 1878]
To install NCOALink directories with the GUI [page 1386]
To install NCOALink directories from the command line [page 1387]

3.5.4.15.1.1.6 DSF2 Certification

DSF2 is a feature that must be purchased before testing.

The DSF2 certification consists of the following steps:

1. Application and Self-Certification Statement Approval
2. Documentation Requirements
3. Stage I Interface Development
4. DSF2 Testing and Certification
5. Execution of License

The entire process is detailed in the USPS *DSF2 Certification Package* document posted on the RIBBS website.

Select the DSF2 Certification Package link on *https://ribbs.usps.gov/dsf2/documents/tech_guides/* .

The *DSF2 Certification Package* contains processes and procedures and the necessary forms for you to complete the five steps listed above.

3.5.4.15.1.1.6.1 DSF2 Equipment Information for USPS certifications

In the DSF2 Certification Package document, there is an Equipment Information form. You are required to provide information about the software you are using to certify for DSF2. Use the information in the following table as you complete the form for the DSF2 certification process.

Equipment Information form	Description
Interface Software Vendor	SAP Americas, Inc.
Interface Software Product Name	ACE
Interface Software Product Version	Contact SAP Business User Support.
Address Matching ZIP+4 Product Name	ACE
Address Matching ZIP+4 Product Version	Contact SAP Business User Support.
Address Matching ZIP+4 System	Closed System
Interface Hardware Vendor/Model/Type	N/A The software is not hardware dependent
Interface Hardware Operating System	N/A The software is not hardware dependent
Interface Hardware Serial Number	N/A The software is not hardware dependent

Find the DSF2 Certification Package document on the RIBBS website at *https://ribbs.usps.gov/dsf2/documents/tech_guides/* .

3.5.4.15.1.1.6.2 DSF2 Certification blueprints

SAP BusinessObjects provides DSF2 certification blueprints for the three types of DSF2 certifications. The blueprints contain the corresponding projects, jobs, data flows, and input/output formats. Additionally, the blueprints can be used to process a test file provided by the USPS during an audit.

Import the DSF2 certification blueprints from **<DS_COMMON_DIR>**`\DataQuality\Certifications`, where **<DS_COMMON_DIR>** is the software's common configuration location.

The table below contains the file names for the USPS DSF2 Augment certification:

Object	Name
ATL file	us_dsf2_certification.atl
Project	DataQualityCertificationDSF2
Job	Job_DqBatchUSAReg_DSF2Augment
data flow	DF_DqBatchUSAReg_DSF2Augment
Input file format	DqUsaDSF2Augment_in
Output file format	DqUsaDSF2Augment_out

The table below contains the file names for the USPS DSF2 Invoice certification:

Project	Name
ATL file	us_dsf2_certification.atl
Project	DataQualityCertificationDSF2
Job	Job_DqBatchUSAReg_DSF2Invoice
data flow	DF_DqBatchUSAReg_DSF2Invoice
Input file format	DqUsaDSF2Invoice_in
Output file format	DqUsaDSF2Invoice_out

The table below contains the file names for the USPS DSF2 Sequence certification:

Project	Name
ATL file	us_dsf2_certification.atl
Project	DataQualityCertificationDSF2
Job	Job_DqBatchUSAReg_DSF2Sequence
data flow	DF_DqBatchUSAReg_DSF2Sequence
Input file format	DqUsaDSF2Sequence_in
Output file format	DqUsaDSF2Sequence_out

3.5.4.15.1.1.6.3 To set up the DSF2 certification blueprints

Before performing the steps below you must import the DSF2 blueprints.

Follow these steps to set up the DSF2 Augment, Invoice, and Sequence certification blueprints.

1. In the Designer, select *Tools* > *Substitution Parameter Configurations* .
   The *Substitution Parameter Editor* opens.
2. Choose the applicable configuration from the *Default Configuration* drop list and enter values for your company's information and reference file locations.

   > **Note**
   >
   > DSF2 Augment only. Remember to enter the static directories location for the $$RefFilesUSPSStatic substitution variable.

3. Open the DataQualityCertificationDSF2 project (downloaded with the blueprint).
4. Expand the desired certification job and data flow. For example, if you are setting up for DSF2 Augment, expand the Job_DqBatchUSAReg_DSF2Augment job and then the DF_DqBatchUSAReg_DSF2Augment data flow.
5. Double-click the applicable input file format (*.in) to open the *Source File Editor*. For example, for DSF2 Augment certification, double-click `DSF2_Augment.in`.
6. In the *Data Files(s)* property group make the following changes:
   a) In the *Root Directory* option, type the path or browse to the directory containing the input file.

If you type the path, do not type a backslash or forward slash at the end of the file path.

b) In the *File name(s)* option, change the input file name to the name of the file provided by the USPS.

7. Double-click the applicable output file format (*.out) to open the Target File Editor. For example, for DSF2 Augment certification, double-click `DSF2_Augment.out`.
8. In the Data Files(s) property group make the following changes:

   a) In the *Root Directory* option, type the path or browse to the directory containing the output file.

   If you type the path, do not type a backslash or forward slash at the end of the file path.

   b) (Optional) In the *File name(s)* option, change the output file name to conform to your company's file naming convention.
9. Click the USARegulatory_AddressCleanse transform to open the Transform Editor and click the *Options* tab.

> i Note
>
> For DSF2 Sequence and Invoice certifications, you will open the DSF2_Walk_Sequencer transform.

10. As necessary, in the Reference Files group, enter the correct path location to the reference files.

    For DSF2 Augment certification, use the $$RefFilesUSPSStatic substitution variable to save time.
11. In the CASS Report Options, update each option that is listed as “CHANGE_THIS” if applicable.

Related Information

To import certification blueprints [page 1382]

DSF2 Certification blueprints [page 1393]

NCOALink blueprints [page 1389]

CASS self-certification blueprint [page 1384]

3.5.4.15.2 Country ISO codes and assignment engines

The table shows which engine (if any) provides address correction. Additionally, it lists the 2-character and 3-character ISO code, the 3-digit ISO code, European Postcode prefix, and the level of assignment. The assignment level is based on the reference data that you own.

Table 166: Table Key

Engine	Assignment Level

Canada = C
Global Address = G
USA = U
	Country = C
	Locality = L
	Primary Name = Pn
	Premise = Pr
	Secondary = S

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Afghanistan	AF	AFG	004		G	C, L
Åland Islands	AX	ALA	248	AX	G	C, L
Albania	AL	ALB	008		G	C, L
Algeria	DZ	DZA	012		G	C, L
American Samoa	AS	ASG	016		U G	C, L, Pn, Pr, S C, L
Andorra	AD	AND	020	AND	G	C, L
Angola	AO	AGO	024		G	C, L
Anguilla	AI	AIA	660		G	C, L
Antarctica	AQ	ATA	010		G	C
Antigua and Barbuda	AG	ATG	028		G	C, L
Argentina	AR	ARG	032		G	C, L
Armenia	AM	ARM	051		G	C, L
Aruba	AW	ABW	533		G	C, L
Australia	AU	AUS	036		G	C, L, Pn, Pr, S
Austria	AT	AUT	040	A	G	C, L, Pn, Pr, S
Azerbaijan	AZ	AZE	031		G	C, L
Bahamas	BS	BHS	044		G	C, L
Bahrain	BH	BHR	048		G	C, L
Bangladesh	BD	BGD	050		G	C, L
Barbados	BB	BRB	052		G	C, L
Belarus	BY	BLR	112		G	C, L
Belgium	BE	BEL	056	B	G	C, L, Pn, Pr
Belize	BZ	BLZ	084		G	C, L
Benin	BJ	BEN	204		G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Bermuda	BM	BMU	060		G	C, L
Bhutan	BT	BTN	064		G	C, L
Bolivia, Plurinational State of	BO	BOL	068		G	C, L
Bonaire, Sint Eustatius, and Saba	BQ	BES	535		G	C, L
Bosnia and Herzegovina	BA	BIH	070		G	C, L
Botswana	BW	BWA	072		G	C, L
Bouvet Island	BV	BVT	074		G	C
Brazil	BR	BRA	076		G	C, L, Pn, Pr
British Indian Ocean Territory	IO	IOT	086		G	C
British Virgin Islands	VG	VGB	092		G	C, L
Brunei Darussalam	BN	BRN	096		G	C, L
Bulgaria	BG	BGR	100	BG	G	C, L
Burkina Faso	BF	BFA	854		G	C, L
Burundi	BI	BDI	108		G	C, L
Cambodia	KH	KHM	116		G	C, L
Cameroon	CM	CMR	120		G	C, L
Canada	CA	CAN	124		C G	C, L, Pn, Pr, S C,L
Cape Verde	CV	CPV	132		G	C
Cayman Islands	KY	CYM	136		G	C

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Central African Republic	CF	CAF	140		G	C, L
Chad	TD	TCD	148		G	C, L
Chile	CL	CHL	152		G	C, L
China	CN	CHN	156		G	C, L, Pn, Pr
Christmas Island (Included in the Australia data package)	CX	CXR	162		G	C, L
Cocos (Keeling) Isles (Included in the Australia data package)	CC	CCK	166		G	C, L
Colombia	CO	COL	170		G	C, L
Comoros	KM	COM	174		G	C, L
Congo, Republic of	CG	COG	178		G	C, L
Congo, Democratic Republic of	CD	COD	180		G	C, L
Cook Islands	CK	COK	184		G	C, L
Costa Rica	CR	CRI	188		G	C, L
Côte d'Ivoire	CI	CIV	384		G	C, L
Croatia (Hrvatska)	HR	HRV	191	HR	G	C, L
Cuba	CU	CUB	192		G	C, L
Curaçao	CW	CUW	531		G	C, L
Cyprus	CY	CYP	196	CY	G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Czech Republic (Czechoslovakia)	CZ	CZE	203	CZ	G	C, L, Pn, Pr
Democratic People's Republic of Korea	KP	PRK	408		G	C, L
Denmark	DK	DNK	208	DK	G	C, L, Pn, Pr
Djibouti	DJ	DJI	262		G	C, L
Dominica	DM	DMA	212		G	C, L
Dominican Republic	DO	DOM	214		G	C, L
Timor-Leste	TL	TLS	626		G	C
Ecuador	EC	ECU	218		G	C, L
Egypt	EG	EGY	818		G	C, L
El Salvador	SV	SLV	222		G	C, L
Equatorial Guinea	GQ	GNQ	226		G	C, L
Eritrea	ER	ERI	232		G	C, L
Estonia	EE	EST	233	EE	G	C, L, Pn, Pr
Ethiopia	ET	ETH	231		G	C, L
Falkland Islands	FK	FLK	238		G	C, L
Faroe Islands (Included in the Denmark data package)	FO	FRO	234	FO	G	C, L, Pn, Pr
Federated States of Micronesia	FM	FSM	583		U G	C, L, Pn, Pr, S C, L
Fiji	FJ	FJI	242		G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Finland	FI	FIN	246	FI	G	C, L, Pn, Pr
France	FR	FRA	250	F	G	C, L, Pn, Pr, S
French Guiana (Included in the France data package)	GF	GUF	254		G	C, L, Pn, Pr
French Polynesia (Included in the France data package)	PF	PYF	258		G	C, L, Pn, Pr
French Southern Territories	TF	ATF	260		G	C, L, Pn, Pr
Gabon	GA	GAB	266		G	C, L
Gambia	GM	GMB	270		G	C, L
Georgia	GE	GEO	268		G	C, L
Germany	DE	DEU	276	D	G	C, L, Pn, Pr
Ghana	GH	GHA	288		G	C, L
Gibraltar	GI	GIB	292		G	C, L
Greece	GR	GRC	300	GR	G	C, L, Pn, Pr
Greenland (Included in the Denmark data package)	GL	GRL	304	GL	G	C, L, Pn, Pr
Grenada	GD	GRD	308		G	C, L
Guadeloupe (Included in the France data package)	GP	GLP	312		G	C, L, Pn, Pr
Guam	GU	GUM	316		U G	C, L, Pn, Pr, S C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Guernsey (Included in the United Kingdom data package)	GG	GGY	831	G	G	C, L, Pn, Pr, S
Guatemala	GT	GTM	320		G	C, L
Guinea	GN	GIN	324		G	C, L
Guinea-Bissau	GW	GNB	624		G	C, L
Guyana	GY	GUY	328		G	C, L
Haiti	HT	HTI	332		G	C, L
Heard Island and McDonald Islands	HM	HMD	334		G	C, L
Holy See (Vatican City State) (Included in the Italy data package)	VA	VAT	336		G	C, L, Pn, Pr
Honduras	HN	HND	340		G	C, L
Hong Kong	HK	HKG	344		G	C, L
Hungary	HU	HUN	348	H	G	C, L, Pn, Pr
Iceland	IS	ISL	352	IS	G	C, L
India	IN	IND	356		G	C, L, Pn, Pr
Indonesia	ID	IDN	360		G	C, L
Iraq	IQ	IRQ	368		G	C, L
Ireland, Republic of	IE	IRL	372	IRL	G	C, L
Islamic Republic of Iran	IR	IRN	364		G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Israel	IL	ISR	376		G	C, L
Isle of Man (Included in the United Kingdom data package)	IM	IMN	833		G	C, L, Pn, Pr, S
Italy	IT	ITA	380	I	G	C, L, Pn, Pr
Jamaica	JM	JAM	388		G	C, L
Japan	JP	JPN	392		G	C, L, Pn, Pr, S
Jersey (Included in the United Kingdom data package)	JE	JEY	832		G	C, L, Pn, Pr, S
Jordan	JO	JOR	400		G	C, L
Kazakhstan	KZ	KAZ	398		G	C, L
Kenya	KE	KEN	404		G	C, L
Kiribati	KI	KIR	296		G	C, L
Kuwait	KW	KWT	414		G	C, L
Kyrgyzstan	KG	KGZ	417		G	C, L
Lao People's Democratic Republic	LA	LAO	418		G	C, L
Latvia	LV	LVA	428	LV	G	C, L, Pn, Pr
Lebanon	LB	LBN	422		G	C, L
Lesotho	LS	LSO	426		G	C, L
Liberia	LR	LBR	430		G	C, L
Libya	LY	LBY	434		G	C, L
Liechtenstein (Included in	LI	LIE	438	FL	G	C, L, Pn, Pr

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
the Switzerland data package)						
Lithuania	LT	LTU	440	LT	G	C, L, Pn, Pr
Luxembourg	LU	LUX	442	L	G	C, L, Pn, Pr
Macao	MO	MAC	446		G	C, L
Macedonia	MK	MKD	807	MK	G	C, L
Madagascar	MG	MDG	450		G	C, L
Malaysia	MY	MYS	458	M	G	C,L
Malawi	MW	MWI	454		G	C, L
Maldives	MV	MDV	462		G	C, L
Mali	ML	MLI	466		G	C, L
Malta	MT	MLT	470		G	C, L
Marshall Islands	MH	MHL	584		U G	C, L, Pn, Pr, S C, L
Martinique (Included in the France data package)	MQ	MTQ	474		G	C, L, Pn, Pr
Mauritania	MR	MRT	478		G	C, L
Mauritius	MU	MUS	480		G	C, L
Mayotte (Included in the France data package)	YT	MYT	175		G	C, L, Pn, Pr
Mexico	MX	MEX	484		G	C, L
Moldova	MD	MDA	498	MD	G	C, L
Monaco (Included in the	MC	MCO	492	F	G	C, L, Pn, Pr

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
France data package)						
Mongolia	MN	MNG	496		G	C, L
Montserrat	MS	MSR	500		G	C, L
Montenegro	ME	MNE	499		G	C, L
Morocco	MA	MAR	504		G	C, L
Mozambique	MZ	MOZ	508		G	C, L
Myanmar	MM	MMR	104		G	C, L
Namibia	NA	NAM	516		G	C, L
Nauru	NR	NRU	520		G	C, L
Nepal	NP	NPL	524		G	C, L
Netherlands	NL	NLD	528	NL	G	C, L, Pn, Pr
New Caledonia (Included in the France data package)	NC	NCL	540		G	C, L, Pn, Pr
New Zealand	NZ	NZL	554		G	C, L, Pn, Pr, S
Nicaragua	NI	NIC	558		G	C, L
Niger	NE	NER	562		G	C, L
Nigeria	NG	NGA	566		G	C, L
Niue	NU	NIU	570		G	C, L
Norfolk Island (Included in the Australia data package)	NF	NFK	574		G	C, L
Northern Mariana Islands	MP	MNP	580		U G	C, L, Pn, Pr, S C, L
Norway	NO	NOR	578	N	G	C, L, Pn, Pr

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Occupied Palestinian Territory	PS	PSE	275		G	C
Oman	OM	OMN	512		G	C, L
Pakistan	PK	PAK	586		G	C, L
Palau	PW	PLW	585		U G	C, L, Pn, Pr, S C, L
Panama	PA	PAN	591		G	C, L
Papua New Guinea	PG	PNG	598		G	C, L
Paraguay	PY	PRY	600		G	C, L
Peru	PE	PER	604		G	C, L
Philippines	PH	PHL	608		G	C, L
Pitcairn	PN	PCN	612		G	C, L
Poland	PL	POL	616	PL	G	C, L, Pn, Pr
Portugal	PT	PRT	620	P	G	C, L, Pn, Pr,S
Province of China Taiwan	TW	TWN	158		G	C, L
Puerto Rico	PR	PRI	630		U G	C, L, Pn, Pr, S C, L
Qatar	QA	QAT	634		G	C, L
Republic of Korea	KR	KOR	410		G	C, L
Réunion (Included in the France data package)	RE	REU	638		G	C, L, Pn, Pr
Romania	RO	ROU	642	RO	G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Russian Federation	RU	RUS	643	RUS	G	C, L
Rwanda	RW	RWA	646		G	C, L
Saint Barthélemy (Included in the France data package)	BL	BLM	652		G	C, L
Saint Helena, Ascension, and Tristan da Cunha	SH	SHN	654		G	C, L
Saint Kitts and Nevis	KN	KNA	659		G	C, L
Saint Lucia	LC	LCA	662		G	C, L
Saint Martin (French part) (Included in the France data package)	MF	MAF	663		G	C, L
Saint Pierre and Miquelon (Included in the France data package)	PM	SPM	666		G	C, L, Pn, Pr
Saint Vincent & Grenadines	VC	VCT	670		G	C, L
Samoa	WS	WSM	882		G	C, L
San Marino (Included in the Italy data package)	SM	SMR	674	SMR	G	C, L, Pn, Pr
Sao Tome and Principe	ST	STP	678		G	C, L
Saudi Arabia	SA	SAU	682		G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Senegal	SN	SEN	686		G	C, L
Serbia	RS	SRB	688		G	C, L
Seychelles	SC	SYC	690		G	C, L
Sierra Leone	SL	SLE	694		G	C, L
Singapore	SG	SGP	702		G	C, L
Sint Maarten (Dutch Part)	SX	SXM	534		G	C, L
Slovakia	SK	SVK	703		G	C, L, Pn, Pr
Slovenia	SI	SVN	705		G	C, L
Solomon Islands	SB	SLB	090		G	C, L
Somalia	SO	SOM	706		G	C, L
South Africa	ZA	ZAF	710		G	C, L
South Georgia and the South Sandwich Islands	GS	SGS	239		G	C, L
South Sudan	SS	SDN	728		G	C, L
Spain	ES	ESP	724	E	G	C, L, Pn, Pr
Sri Lanka	LK	LKA	144		G	C, L
Sudan	SD	SDN	736		G	C, L
Suriname	SR	SUR	740		G	C, L
Svalbard and Jan Mayen (Included in the Norway data package)	SJ	SJM	744		G	C
Swaziland	SZ	SWZ	748		G	C, L
Sweden	SE	SWE	752	S	G	C, L, Pn, Pr

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Switzerland	CH	CHE	756	CH	G	C, L, Pn, Pr
Syrian Arab Republic	SY	SYR	760		G	C, L
Tajikistan	TJ	TJK	762		G	C, L
Thailand	TH	THA	764		G	C, L
Togo	TG	TGO	768		G	C, L
Tokelau	TK	TKL	772		G	C, L
Tonga	TO	TON	776		G	C, L
Trinidad and Tobago	TT	TTO	780		G	C, L
Tunisia	TN	TUN	788	TN	G	C, L
Turkey	TR	TUR	792	TR	G	C, L, Pn, Pr
Turkmenistan	TM	TKM	795		G	C, L
Turks and Caicos Islands	TC	TCA	796		G	C, L
Tuvalu	TV	TUV	798		G	C, L
Uganda	UG	UGA	800		G	C, L
Ukraine	UA	UKR	804	UK	G	C, L
United Arab Emirates	AE	ARE	784		G	C, L
United Kingdom	GB	GBR	826	GB	G	C, L, Pn, Pr, S
United Republic of Tanzania	TZ	TZA	834		G	C, L
United States	US	USA	840		U G	C, L, Pn, Pr, S C, L
United States Minor Outlying Islands	UM	UMI	581		U G	C, L, Pn, Pr, S

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
U.S. Virgin Islands	VI	VIR	850		U G	C, L, Pn, Pr, S
Uruguay	UY	URY	858		G	C, L
Uzbekistan	UZ	UZB	860		G	C, L
Vanuatu	VU	VUT	548		G	C, L
Venezuela	VE	VEN	862		G	C, L
Viet Nam	VN	VNM	704		G	C, L
Wallis and Futuna	WF	WLF	876		G	C, L, Pn, Pr
Western Sahara	EH	ESH	732		G	C, L
Yemen	YE	YEM	887		G	C, L
Zambia	ZM	ZMB	894		G	C, L
Zimbabwe	ZW	ZWE	716		G	C, L

3.5.4.15.3 Information codes (Global Address Cleanse)

Information codes are four characters that explain why an address is unassigned. Information codes have six levels of classification:

- The 1000 level represents input record discrepancies.
- The 2000 level represents inconsistent last line information.
- The 3000 level represents inconsistent address information.
- The 4000 level represents inconsistent secondary address information.
- The 5000 level represents all other types of information.
- The 6000 level represents an unclassified error.

The table also shows that each information code is available based on the engine(s) that you enable.

- Canada (C)
- Global Address (G)
- USA (U)
- All engines: Consists of C, G, and U.
- Transform Level (T): Information code does not come from a specific engine.

Use the following table to determine the code assigned to the Info_Code output field.

Information code	Description	Engine(s)
1020	Address validated in multiple countries.	T
1030	No country found by Country ID or no country set for the record.	T
1040	Address contains at least one character that is not part of the character set supported by the engine.	T
1060	The country identified is not supported by any of the active engines.	T
1080	The script identified is not supported by any of the active engines.	T
2000	Unable to identify locality, region, and/or postcode information on input.	All engines
2010	Unable to identify locality and invalid postcode found.	All engines
2020	Unable to identify postcode. Invalid locality is preventing a possible address correction.	All engines
2030	Invalid locality and postcode are preventing a possible address correction.	All engines
2040	Invalid postcode is preventing a locality selection.	G, U
2050	Lastline matches are too close to choose one.	G
3000	Locality, region, and postcode are valid. Unable to identify the primary address line.	All engines
3010	Locality, region, and postcode are valid. Unable to match primary name to directory.	All engines
3020	Possible primary name matches are too close to choose one.	All engines
3030	Primary range is missing on input or not in the directory.	All engines
3050	An invalid or missing primary type is preventing a possible address match.	All engines
3060	A missing primary type and prefix/postfix (directional) is preventing a possible address match.	G, U

Information code	Description	Engine(s)
3070	An invalid or missing prefix/postfix (directional) is preventing a possible address match.	All engines
3080	An invalid or missing postcode is preventing a possible address match.	All engines
3090	An invalid or missing locality is preventing a possible address match.	G, U
3100	Possible address-line matches are too close to choose one.	All engines
3110	Address conflicts with postcode and the same primary name has a different postcode.	C
3200	The building name is missing on input or not in the directory.	G
3210	The building's address is not in the directory.	G
3220	Possible building names are too close to choose one.	G
3250	The building name is missing on input or not in the directory, or the range is missing on input or invalid with the input building.	G
3300	The postcode only lookup returned multiple primary names.	G
4000	The secondary information is missing on input or not in the directory.	All engines
4010	Possible secondary address line matches are too close to choose one.	All engines
4500	The organization is missing on input or not in the directory.	G
4510	The organization's address is not in the directory.	G
4520	Possible organization names are too close to choose one.	G
5000	The address was valid, but the postal authority classified this address as undeliverable.	G, U
5010	The address does not reside in the specified country.	C, U

Information code	Description	Engine(s)
5020	The entire input record was blank.	T
5030	The country's postal authority will not permit assignment due to violation of an assignment rule.	G
5040	Assignment not made after lastline processing due to a violation of lastline assignment rules.	G
6000	Unclassified error.	All engines

3.5.4.15.4 Status codes (Global Address Cleanse)

Status codes (assigned to the Status_Code output field) are five or six characters that represent the corrections made to the address during processing. The number of characters depends on the engine used for processing.

- The first character is always an S (for Status).
- The second character is associated with any last line corrections.
- The third character is associated with any address line corrections.
- The fourth character is associated with any secondary address line corrections.
- The fifth character is associated with changes to components that are not considered basic address components (Other Primary Address and Other Secondary Address).
- The sixth component indicates additional information about a record that is not related to a change in the address.

Second character

The value of the second character depends on corrections to the country, postcode, region, or locality.

Value	Description
0	No significant difference between the input data and the corrected data.
1	Corrected country.
2	Corrected postal code.
3	Corrected country and postal code.
4	Corrected region.
5	Corrected country and region.

Value	Description
6	Corrected postal code and region.
7	Corrected country, postal code, and region.
8	Corrected locality.
9	Corrected country and locality.
A	Corrected postal code and locality.
B	Corrected country, postal code, and locality.
C	Corrected region and locality.
D	Corrected country, region, and locality.
E	Corrected postal code, region, and locality.
F	Corrected country, postal code, region, and locality.

Third character

The value of the third character depends on corrections to the pre/post directionals, primary type, primary name, and primary range.

Value	Description
0	No significant difference between the input data and the corrected data.
1	Corrected pre/post directional.
2	Corrected primary type.
3	Corrected pre/post directional and primary type.
4	Corrected primary name.
5	Corrected pre/post directional and primary name.
6	Corrected primary type and primary name.
7	Corrected pre/post directional, primary type, and primary name.
8	Corrected primary range.
9	Corrected pre/post directional and primary range.

Value	Description
A	Corrected primary type and primary range.
B	Corrected pre/post directional, primary type, and primary range.
C	Corrected primary name and primary range.
D	Corrected pre/post directional, primary name, and primary range.
E	Corrected primary type, primary name, and primary range.
F	Corrected pre/post directional, primary type, primary name, and primary range.

Fourth character

The value of the fourth character depends on corrections to the unit description, unit number, firm, building name, floor description, floor number, stairwell description, stairwell name, Wing description, and Wing name.

Value	Description
0	No significant difference between the input data and the corrected data.
1	Corrected one or more secondary address component (unit description, floor description, stairwell description, or wing description).
2	Corrected one or more secondary address component (unit number, floor number, stairwell name, or wing name).
3	Corrected one or more secondary address component (unit description, unit number, floor description, floor number, stairwell description, stairwell name, wing description, or wing name).
4	Corrected building name.
5	Corrected one or more secondary address component (unit description, floor description, stairwell description, or wing description), and building name.
6	Corrected the one or more secondary address component (unit number, floor number, stairwell name, or wing name), and building name.
7	Corrected one or more secondary address component (unit description, unit number, floor description, floor number, stairwell description, stairwell name, wing description, or wing name), and building name.
8	Corrected firm.

Value	Description
9	Corrected one or more secondary address component (unit description, floor description, stairwell description, or wing description), and firm.
A	Corrected one or more secondary address component (unit number, floor number, stairwell name, or wing name), and firm.
B	Corrected one or more secondary address component (unit description, unit number, floor description, floor number, stairwell description, stairwell name, wing description, or wing name), and firm.
C	Corrected building name and firm.
D	Corrected one or more secondary address component (unit description, floor description, stairwell description, or wing description), building name, and firm.
E	Corrected one or more secondary address component (unit number, floor number, stairwell name, or wing name), building name, and firm.
F	Corrected one or more secondary address component (unit description, unit number, floor description, floor number, stairwell description, stairwell name, wing description, or wing name), building name, and firm.

Fifth Character

The value of the fifth character depends on changes to components that are not considered basic address components (Other Primary Address and Other Secondary Address).

Other Primary Address components:

- Primary_Delivery_Mode
- Primary_Delivery_Number

Other Secondary Address components:

- Delivery_Installation_Name
- Delivery_Installation_Qualifier
- Delivery_Installation_Type

Value	Description
0	No significant change between the input data and the corrected data.
1	Changed the Other Primary Address components.
2	Changed the Other Secondary Address components.
3	Changed the Other Primary Address and Other Secondary Address components.

Value	Description
4	Changed the point of reference.
5	Changed the Other Primary Address components and the point of reference.
6	Changed the Other Secondary Address components and the point of reference.
7	Changed the Other Primary Address, Other Secondary Address components, and the point of reference.

Sixth Character

The value of the sixth character indicates additional information about a record that is not related to a change in the address.

Value	Description
A	Alias record used for assignment. Global Address engine.
B	Base record assignment. Global Address engine (New Zealand). A Bordering Locality. Global Address engine (Australia).
C	An Alias and a Bordering locality. Global Address engine (Australia).
D	Deleted record. Global Address engine (Austria and Germany).
I	Record ignored. Global Address engine (New Zealand).
L	Large Volume Receiver (LVR). Global Address engine (Brazil).
U	Unique address. Global Address engine (New Zealand).

3.5.4.15.5 Quality codes (Global Address Cleanse)

Quality codes relay additional information about the quality of the address. There are six levels of quality codes based on these factors:

- The country of the input data
- The engine used for processing
- The information code
- The status code if there is not an information code

Use the following table to determine the code assigned to the Quality_Code output field.

Quality code	Description
Q1	Perfect address on input. All address components were validated without corrections.
Q2	Corrected address. All address components were validated after corrections were made.
Q3	Not all components of the address could be fully validated. There was insufficient information to make a final correction. However, the assessment of the record leads to the assumption that there is a "high" likelihood that this address is deliverable.
Q4	Not all components of the address could be fully validated. There was insufficient information to make a final correction. However, the assessment of the record leads to the assumption that there is a "fair" likelihood that this address is deliverable.
Q5	Not all components of the address could be fully validated. There was insufficient information to make a final correction. However, the assessment of the record leads to the assumption that there is a "small" likelihood that this address is deliverable.
Q6	Not all components of the address could be fully validated. There was insufficient information to make a final correction. However, the assessment of the record leads to the assumption that it is "highly unlikely" that this address is deliverable.

3.5.4.15.6 Fault codes (USA Regulatory Address Cleanse)

When the transform cannot assign an address, it creates a fault code (Fault_Code output field). This code tells you why the address could not be assigned.

Fault Code	Description
E101	Last line is bad or missing.
E212	No locality and bad postal code.
E213	Bad locality, valid region, and no postal code.
E214	Bad locality and bad postal code.
E216	Bad postal code and can't determine which locality match to select.
E302	No primary address line parsed.
E412	Primary name not found in directory.
E413	Possible primary name matches are too close to choose one.
E420	Primary range is missing.
E421	Primary range is invalid for the street/route/building.
E422	Primary prefix needed, input is wrong or missing.
E423	Primary type needed, input is wrong or missing.
E425	Primary type and directional needed, input is wrong or missing.

Fault Code	Description
E427	Primary postfix needed, input is wrong or missing.
E428	Bad postal code, can't select an address match.
E429	Bad locality, can't select an address match.
E430	Possible address-line matches too close to choose one.
E431	Locality2 needed, input is wrong or missing.
E439	Exact match made in EWS directory.
E500	Other error.
E501	Foreign address.
E502	Input record entirely blank.
E503	Postal code not in area covered by partial USPS directory.
E504	Overlapping ranges in USPS directory.
E505	Address does not exist in the USPS directories. Undeliverable address.
E600	Marked by USPS as unsuitable for delivery of mail.
E601	The primary address number did not DPV confirm, and the Postcode2 was removed.

3.5.4.15.7 Status codes (USA Regulatory Address Cleanse)

When the transform assigns an address, it creates a status code (Status_Code output field). This code can tell you how the input address differs from the assigned address.

Digit	Description
1st	*A*: The transform truncated the address line to make it fit your field. *B*: The transform truncated both the address line and the Locality1_Name. *C*: The transform truncated the Locality1_Name to make it fit your field. *S*: No truncation occurred.
2nd	*0*: Regarding the Locality1_Name, Region1, Postcode1, and Postcode2, there is no significant difference between the input data and the data that the transform assigned. *1*: The transform assigned a different Postcode1. *2*: The transform assigned a different Locality1_Name. *3*: The transform assigned a different Locality1_Name and Postcode1. *4*: The transform assigned a different Region1. *5*: The transform assigned a different Region1 and Postcode1.

Digit	Description
	6: The transform assigned a different Locality1_Name and Region1. 7: The transform assigned a different Locality1_Name, Region1, and Postcode1. 8: The transform assigned a different Postcode2. 9: The transform assigned a different Postcode1 and Postcode2. A: The transform assigned a different Locality1_Name and Postcode2. B: The transform assigned a different Locality1_Name, Postcode1, and Postcode2. C: The transform assigned a different Region1 and Postcode2. D: The transform assigned a different Region1, Postcode1, and Postcode2. E: The transform assigned a different Locality1_Name, Region1, and Postcode2. F: The transform assigned a different Locality1_Name, Region1, Postcode1, and Postcode2.
3rd	0: Regarding the primary name, primary prefix/postfix, and primary type, there is no significant difference between the input and what the transform assigned. 1: The transform assigned a different primary type. 2: The transform assigned a different primary prefix. 3: The transform assigned a different primary prefix and primary type. 4: The transform assigned a different primary postfix. 5: The transform assigned a different primary type and primary postfix. 6: The transform assigned a different primary prefix and primary postfix. 7: The transform assigned a different primary prefix, primary type, and primary postfix. 8: The transform assigned a different primary name. 9: The transform assigned a different primary name and primary type. A: The transform assigned a different primary prefix and primary name. B: The transform assigned a different primary prefix, primary name, and primary type. C: The transform assigned a different primary name and primary postfix. D: The transform assigned a different primary name, primary type, and primary postfix. E: The transform assigned a different primary prefix, primary name, and primary postfix. F: The transform assigned a different primary prefix, primary name, primary postfix, and primary type.
4th	0: Regarding the county number, sort code route, delivery point, and unit description, there is no significant difference between the input data and the data that the transform assigned. 1: The transform assigned a different unit description. 2: The transform assigned a different delivery point.

Digit	Description
	3: The transform assigned a different delivery point and unit description. 4: The transform assigned a different sort code route. 5: The transform assigned a different sort code route and unit description. 6: The transform assigned a different sort code route and delivery point. 7: The transform assigned a different sort code route, delivery point, and unit description. 8: The transform assigned a different county number. 9: The transform assigned a different county number and unit description. A: The transform assigned a different county number and delivery point. B: The transform assigned a different county number, delivery point, and unit description. C: The transform assigned a different county number and sort code route. D: The transform assigned a different county number, sort code route, and unit description. E: The transform assigned a different county number, sort code route, and delivery point. F: The transform assigned a different county number, sort code route, delivery point, and unit description.
5th	0: Regarding the LOT, LOT_Order, and Locality2_Official, there is no significant difference between the input data and the data that the transform assigned. 1: The transform assigned a different LOT. 2: The transform assigned a different LOT_Order. 3: The transform assigned a different LOT and LOT_Order. 4: The transform assigned a different Locality2_Official. 5: The transform assigned a different Locality2_Official and LOT. 6: The transform assigned a different Locality2_Official and LOT_Order. 7: The transform assigned a different Locality2_Official, LOT, and LOT_Order.
6th	Always outputs a zero (0).

3.5.4.15.8 About ShowA and ShowL (USA)

The Show programs are used for looking inside the postal directories to find answers to questions like these:

- Why did the transform standardized the address in an unexpected way?
- Why didn't the transform assign the address?
- Why did the transform’s error code indicate a flaw in the directory?

Note

Run the ShowA/ShowL utilities from a DOS command line using specific command-line options. These options are listed when you enter the following command:

Windows: `showa /op`

UNIX: `showa -op`

Query the postal directories

You can use ShowA to display or output information from the Address_1_Directory, and you can use ShowL to query the City_Directory and the Post_Code_Directory.

Edit configuration files

Each Show utility has its own configuration file. These files contain parameters for controlling how the program behaves.

USA addresses:

Utility	Executable	File name	Location
ShowA	`showa.exe`	`showa.cfg`	**`<LINK_DIR>`** `\dataquality\urac`
ShowL	`showl.exe`	`showl.cfg`	**`<LINK_DIR>`** `\dataquality\urac`

Before you run the Show utilities, set both configuration files for the appropriate country directory. The configuration files contain instructions and detailed information about how to run the programs.

Note

Run the ShowA/ShowL utilities in the same directory as the ShowA/ShowL configuration files. You can change the location of the Show A/L executable files, however the utilities will not run if you did not accept the default location for the configuration files.

Related Information

USA ShowA command line options [page 1422]

3.5.4.15.8.1 ShowA/ShowL modes of operation

The ShowA and ShowL utilities have two modes for entering queries: prompts and command line options.

Prompts

If you type the ShowA/ShowL command without any options at all, then ShowA/ShowL prompts you to enter your query data. It takes all other information and options from the configuration file.

After your first query, ShowA/ShowL prompts you to enter the next one. You can exit by typing "quit" at any prompt.

Command line options

You may enter your query data on your ShowA/ShowL command line. ShowA/ShowL performs one query, displays and/or outputs the results, and then exits.

From the configuration file, ShowA/ShowL takes information about the auxiliary files, output file, display and search options. If you must override any value taken from the configuration file, you can do so. You can use command line options selectively to override where you need to, and depend on the configuration file for the rest. The only value you cannot specify through command- line options is the optional output fields.

> **Note**
>
> If any command option is present, then ShowA/ShowL detects that you are operating in command line mode. It will not prompt you to enter your query data. Use command line options to enter your query data.

3.5.4.15.8.2 USA ShowA command line options

To view a summary of command line options, use this command:

Windows: `showa /op`

UNIX: `showa -op`

The following table lists the command line options and the command descriptions.

UNIX	Windows	Description
-a	/a	Appends information to the output file (if it already exists).
-alias	/alias	Includes preferred alias address lines.
-d	/d	Displays your query data on screen.

UNIX	Windows	Description
-fin	/fin	Expands the query to cover USPS finance area.
-op	/op	Displays the list of options (in this table).
-p	/p	Pauses screen display every 22 lines.
-2:dpbc	/2:dpbc	Enter the DPBC code for `dpbc`.
-4:zip4	/4:zip4	Enter the postcode2 for`zip4`.
-ad:file	/ad:file	Enter the Address-line dictionary and path name (**addrln.dct**) for `file`.
-c:cart	/c:cart	Enter the carrier route number for `cart`.
-f:file	/f:file	Enter the file path and name of the output file (to hold the information from the query instead of just displaying it on screen) for `file`.
-nd:file	/nd:file	Enter the National ZIP+4 directory path and name (**zip4us.dir**) for `file`.
-pre:dir	/pre:dir	Enter the primary prefix (N, NE, E, SE, S, SW, W, NW) for `dir`.
-pos:dir	/pos:dir	Enter the primary postfix (N, NE, E, SE, S, SW, W, NW) for `dir`.
-s:street	/s:street	Enter the street primary name (in quotes if multiple words) for `street`.
-sfx:suffix	/sfx:suffix	Enter the primary type (Ave, Blvd, St, Rd, and so on) for `suffix`.
-sh:range	/sh:range	Enter the street (primary) range high for `range`.
-sl:range	/sl:range	Enter the street (primary) range low or exact for `range`.
-t:type	/t:type	Enter the file type (dBASE3, ASCII, or DELIMITED) for `type`.
-u:urb idx	/u:urb idx	Enter the urbanization Index for `urb idx`.
-z:lo-hi	/z:lo-hi	Enter the low and high range for postcode1 for `lo-hi`.
-z:zip	/z:zip	Enter the postcode1 for `zip`.

Related Information

ShowA/ShowL modes of operation [page 1422]

3.5.4.15.8.3 USA ShowL command line options

To view a summary of command line options, use this command:

Windows: `showl /op`

UNIX: `showl -op`

The following table lists the command line option and the command descriptions.

UNIX	Windows	Description
-a	/a	Appends query information to an output file (if it already exists).
-d	/d	Displays query data on screen.
-op	/op	Displays this list of options.
-p	/p	Pauses screen display every 22 lines.
-ab:query	/ab:query	Enter the abbreviated locality1 for `query`.
-cd:file	/cd:file	Enter the City directory path and name (**city04.dir**) for `file`.
-cn:city	/cn:city	Enter locality1 name (in quotes if multiple words) for `city`.
-dr:dir	/dr:dir	Enter which directory to search, City or ZCF for `dir.`
-f:file	/f:file	Enter the output file path and name for `file`.
-ml:query	/ml:query	Enter the military postcode1 for `query`.
-mz:query	/mz:query	Enter the multi-zone locality1 for `query.`
-pn:query	/pn:query	Enter a place name for `query.`
-st:state	/st:state	Enter region1 (for US use USPS abbreviations or full state names) for `state`.
-t:type	/t:type	Enter the output file type (dBASE3, ASCII, or DELIMITED) for `type`.
-un:query	/un:query	Enter the unique postcode1 for `query`.
-z:zip	/z:zip	Enter the postcode1 for `zip`.
-zd:file	/zd:file	Enter the ZCF directory path and name (**zcf04.dir**) for `file`.
-z:lo-hi	/z:lo-hi	Enter the postcode1 range for `lo-hi`.

Related Information

ShowA/ShowL modes of operation [page 1422]

3.5.4.16 Data Cleanse reference

This section describes the following reference information for use with the Data Cleanse transform:

- Status codes
- Information codes
- Diacritical character conversion chart

3.5.4.16.1 Status codes (Data Cleanse)

Status codes (assigned to the Status_Code output field) have the same format: **`<parser name>`**_**`<output field name>`**_STD or **`<parser name>`**_**`<output field name>`**_ADD. The **`<parser name>`** includes one of the following parsers: custom, date, firm, person, phone, and SSN.

Only the status codes that have standardized output are shown. The status codes show one of the following depending on whether the standardization options are chosen:

- how the data was standardized
- that a specific standard could be used on the data

i Note

When only minor changes occur, a status code may not be generated. See the exceptions at the bottom of this topic.

Person parser

Person parser status codes have the PERSON#_**`<output field>`**_STD or PERSON#**`<output field>`**_ADD format where the pound sign (#) is replaced with the person number, such as Person1 or Person2 through Person6.

Table 167: Person parser status codes

Status code	Description
Person#_Name_Designator_Std	Used a Name_Designator standard.
Person#_Prename_Add	Prename was assigned.
Person#_Prename _Std	Prename was provided and a prename standard was used.

Status code	Description
Person#_Given_Name1_Std	Used a Given_Name1 standard.
Person#_Given_Name2_Std	Used a Given_Name2 standard.
Person#_Family_Name1_Std	Used a Family_Name1 standard.
Person#_Family_Name2_Std	Used a Family_Name2 standard.
Person#_Honorary_Postname_Std	Used an Honorary_Postname standard.
Person#_Maturity_Postname_Std	Used a Maturity_Postname standard.
Person#_Title_Std	Used a Title standard.
Person#_Name_Special_Std	Used a Name_Special standard.

Example

Multiple status codes can be assigned per record and are separated by a comma. For example, the status code:

```
PERSON1_PRENAME_STD,PERSON1_GIVEN_NAME1_STD,PERSON1_FAMILY_NAME1_STD,PERSON2_GIVEN
_NAME2_STD,PERSON6_MATURITY_POSTNAME_STD
```

provides the following information:

- The Person1 parse used standards for Prename, Given_Name1, and Family_Name1 output fields.
- The Person2 parse used standards for the Given_Name2 output field.
- The Person6 parse used standards for the Maturity_Postname output field.

Firm parser

Firm parser status codes have the FIRM#_**`<output field>`**_STD format where the pound sign (#) is replaced with the Firm number, such as Firm1 or Firm2 through Firm6.

Table 168: Firm parser status codes

Status code	Description
Firm#_Firm_Std	Used a Firm standard.
Firm#_Firm_Location_Std	Used a Firm_Location standard.

Example

Multiple status codes can be assigned per record and are separated by a comma. For example, the status code:

```
FIRM1_FIRM_STD,FIRM2_FIRM_LOCATION_STD,FIRM3_FIRM_STD,FIRM3_FIRM_LOCATION_STD
```

provides the following information:

- The Firm1 parse used a Firm standard.
- The Firm2 parse used a Firm_Location standard.

- The Firm3 parse used both a Firm standard and a Firm_Location standard.

Date, Phone and Social Security number parsers

Date, Phone and Social Security number (SSN) parser status codes have the following status code formats:

Table 169: Date, Phone and Social Security number parser status codes

Status code	Description
Date#_Date_Std	Used a Date standard.
Phone#_Phone_Std	Used a Phone number standard.
SSN#_SSN_Std	Used a Social Security number standard.

Example

Multiple status codes can be assigned per record and are separated by a comma. For example, the Date status code, `DATE1_DATE_STD,DATE6_DATE_STD`, shows that both Date1 and Date6 were standardized.

`SSN1_SSN_STD,SSN5_SSN_STD` shows that both SSN1 and SSN5 were standardized.

`PHONE1_PHONE_STD,PHONE3_PHONE_STD` shows that both Phone1 and Phone3 were standardized.

Custom parser

Custom parser status codes have the **<parser name>**#_**<output field>**_STD format where the pound sign (#) is replaced with the parser number.

Table 170: Custom parser status code

Status code	Description
<parser name>#_**<output field>**_Std	Used the **<output field>**'s standard.

Example

Multiple status codes can be assigned per record and are separated by a comma. For example, the Automobile status code, `AUTOMOBILE1_COLOR_STD,AUTOMOBILE1_SIZE_STD,AUTOMOBILE2_COLOR_STD`, shows the following information:

- Automobile1 used a color and size standard.
- Automobile2 used a color standard.

Status code exceptions

When there are only minor changes, a status code may not be generated. There are several instances where a status code is not generated.

Exception	Description	Example
Trailing period	A status code is not generated when a period is added or removed after a piece of data.	J Smith has a standard of "J." No status code. J. Smith has a standard of "J" No status code. UK has a standard of "U.K." A status code is generated because there are more punctuation changes than the trailing period.
Capitalization option	A status code is not generated when the only difference between the standard and the original data is how the data is cased.	Mcdonald has a standard of "McDonald" but is output as "MCDONALD" because the *Capitalization* option is set to *Upper*. No status code.
Remove punctuation option	A status code is not generated when the only difference between the standard and the original data is removed punctuation when this option is set to *Yes*.	UK has a standard of "U.K." However, the *Remove punctuation* option is set to *Yes*. No status code.
Remove diacritical characters option	A status code is not generated when the only difference between the standard and the original data is the diacritical characters when this option is set to *Yes*.	Hernandez has a standard of "Hernández". However, the *Remove diacritical characters* option is set to *Yes*. No status code. Hernandez has a standard of "Hernández". However, the *Remove diacritical characters* option is set to *No*. A status code is generated to show that a standard was used.

3.5.4.16.2 Information codes (Data Cleanse)

Information codes (assigned to the Info_Code output field) provide information about the data that may be suspect and might require a manual review. The output field contains one or more codes separated by a comma. Each code begins with a letter followed by three numbers classified as follows:

- Date parse information

Information code format	Description
D1##	Date1 parse level information
D2##	Date2 parse level information
D3##	Date3 parse level information
D4##	Date4 parse level information
D5##	Date5 parse level information

Information code format	Description
D6##	Date6 parse level information

- Firm parse information

Information code format	Description
F1##	Firm1 parse level information
F2##	Firm2 parse level information
F3##	Firm3 parse level information
F4##	Firm4 parse level information
F5##	Firm5 parse level information
F6##	Firm6 parse level information

- Input field information

Information code format	Description
I###	Input field level information

- Person parse information

Information code format	Description
P1##	Person1 parse level information
P2##	Person2 parse level information
P3##	Person3 parse level information
P4##	Person4 parse level information
P5##	Person5 parse level information
P6##	Person6 parse level information

- Phone parse information

Information code format	Description
T1##	Phone1 parse level information
T2##	Phone2 parse level information
T3##	Phone3 parse level information
T4##	Phone4 parse level information
T5##	Phone5 parse level information
T6##	Phone6 parse level information

- Record level information

Information code format	Description
R###	Record level information

The following table shows more details about the information codes.

Information code	Description
R001	All input field data went to one or more Extra output fields; nothing was parsed for the record.
R002	Parsed some of the input fields. One or more other input fields went to the Extra output field.
R003	Parsed part of the input fields. Some of the record data went to the Extra output field.
R004	All of the record's input fields were parsed into appropriate output fields. None of the input data went to the Extra output field.
R400	The data in Option_Content_Domain_Sequence overrode the content domain sequence transform option.
R405	The data in Option_Content_Domain_Sequence was not recognized as a content domain sequence. The data in Option_Country or the data in content domain sequence transform option was used to determine the content domain sequence.
R410	The data in Option_Output_Format overrode the output format data cleanse transform option.
R415	The data in Option_Output_Format was not recognized as a valid output format. The data in Option_Country or the output format transform option was used to determine the output format.
R420	The data in Option_Country was recognized as an ISO2 country code and overrode the content domain sequence transform option.
R421	The data in Option_Country was recognized as an ISO2 country code and overrode the output format transform option.
R425	The data in Option_Country was not recognized as an ISO2 country code. One or more situations occurred where Option_Country data was used to override the content domain sequence and/or output format transform options, but failed: ● Attempted to use Option_Country to override the content domain sequence transform option. This occurs when the Option_Content_Domain_Sequence data is invalid (R405) or the data is not supplied (no status code is generated when this occurs). The data was parsed based using the content domain sequence transform option. ● Attempted to use Option_Country to override the output transform option. This occurs when the Option_Output_Format data is invalid (R415) or is not supplied (no status code is generated when this occurs). The data was parsed based using the output format transform option.
P#01	The person# parse contained some data that was not found in the cleansing package. This information code does not report on title information, which is different than information code P#51.
P#02	The person# parse had a close firm parse. This is only applicable for Person_Firm multiline parse when using the Person_Firm multiline parser or Name_Firm_Line when the data came from the input field Name_Firm_Line.
P#03	The person# parse was a presumptive name parse (based on reasonable evidence).
P#04	The person# parse has no given name, or has a questionable given name.

Information code	Description
P#05	The person# parse has no family name, or has a questionable family name.
P#51	The person# parse contained a title token that was not found in the cleansing package. This is different than information code P#01.
F#01	The firm# parse contained some data that was not found in the cleansing package.
F#02	The firm# parse had a close person parse. This is only applicable for Person_Firm multiline parse when using the Person_Firm multiline parser or Name_Firm_Line when the data came from the input field Name_Firm_Line.
F#03	The firm# parse was a presumptive firm parse (based on reasonable evidence).
D#01	Date# was not in the expected format.
D#02	Date# was converted from 2-digits to 4-digits. The century threshold transform option was applied.
D#03	Date# was in the Day_Month_Year format.
D#04	Date# was in the Month_Day_Year format.
D#05	Date# was in the Year_Month_Day format.
D#06	Date# was in the Year_Day_Month format.
D#07	Date# was in an ambiguous format. There is more than one possible format for the date. For example, 12/09/10 is valid for all of the formats, whereas 03/16/94 is valid only for the Month_Day_Year format.
T#01	The phone# parse did not have a North American area code.
I111-I116	All of the input data in Name_Line# went to one or more Extra output fields; nothing was parsed for this input field.
I131-I136	Parsed some of the input data in Name_Line#, the rest of the data is in the Extra output field.
I151-I156	All of the input data in Title_Line# went to one or more Extra output fields; nothing was parsed for this input field.
I171-I176	Parsed some of the input data in Title_Line#, the rest of the data is in the Extra output field.
I311-I316	All of the input data in Name_Or_Firm_Line# went to one or more Extra output fields; nothing was parsed for this input field.
I331-I336	Parsed some of the input data in Name_Or_Firm_Line#, the rest of the data is in the Extra output field.
I351-I352	All of the input data in Firm_Line# went to one or more Extra output fields; nothing was parsed for this input field.
I371-I372	Parsed some of the input data in Firm_Line#, the rest of the data is in the Extra output field.
I511-I516	All of the input data in Date# went to one or more Extra output fields; nothing was parsed for this input field.

Information code	Description
I531-I536	Parsed some of the input data in Date#, the rest of the data is in the Extra output field.
I711-I716	All of the input data in Email# went to one or more Extra output fields; nothing was parsed for this input field.
I731-I736	Parsed some of the input data in Email#, the rest of the data is in the Extra output field.
I751-I756	All of the input data in Phone# went to one or more Extra output fields; nothing was parsed for this input field.
I771-I776	Parsed some of the input data in Phone#, the rest of the data is in the Extra output field.
I811-I816	All of the input data in SSN# went to one or more Extra output fields; nothing was parsed for this input field.
I831-I836	Parsed some of the input data in SSN#, the rest of the data is in the Extra output field.
I851-I856	All of the input data in UDPM# went to one or more Extra output fields; nothing was parsed for this input field.
I871-I876	Parsed some of the input data in UDPM#, the rest of the data is in the Extra output field.
I901-I912	All of the input data in Date# went to one or more Extra output fields; nothing was parsed for this input field.
I951-I962	Parsed some of the input data in Multiline#, the rest of the data is in the Extra output field.

3.5.4.16.3 Diacritical character conversion chart

In the Data Cleanse transform, there is an option where you can remove the diacritical characters from the data on output. The following table shows the diacritical characters and the ASCII equivalent letter that replaces the diacritical character.

Note

The diacritical character is listed first, separated by a comma and then the ASCII replacement letter.

Table 171: Diacritical characters conversion chart

À,A	Ĕ,E	Ļ,L	Ś,S	Ź,Z	ē,e	ľ,l	ŝ,s
Á,A	Ė,E	Ľ,L	Ŝ,S	Ż,Z	ĕ,e	ŀ,l	ş,s
Â,A	Ę,E	Ŀ,L	Ş,S	à,a	ė,e	ł,l	ţ,t
Ã,A	Ĝ,G	Ł,L	Ţ,T	á,a	ę,e	ñ,n	ť,t
Ä,AE	Ğ,G	Ñ,N	Ť,T	â,a	ĝ,g	ń,n	ŧ,t
Å,A	Ġ,G	Ń,N	Ŧ,T	ã,a	ğ,g	ņ,n	ù,u
Ā,A	Ģ,G	Ņ,N	Ù,U	ä,ae	ġ,g	ň,n	ú,u
Ă,A	Ĥ,H	Ň,N	Ú,U	å,a	ģ,g	ò,o	û,u

Ą,A	Ħ,H	Ò,O	Û,U	ā,a	ĥ,h	ó,o	ü,ue
Ç,C	Ì,I	Ó,O	Ü,UE	ă,a	ħ,h	ô,o	ũ,u
Ć,C	Í,I	Ô,O	Ũ,U	ą,a	ì,i	õ,o	ū,u
Ĉ,C	Î,I	Õ,O	Ū,U	ç,c	í,i	ö,oe	ŭ,u
Ċ,C	Ï,I	Ö,OE	Ŭ,U	ć,c	î,i	ø,o	ů,u
Č,C	Ĩ,I	Ø,O	Ů,U	ĉ,c	ï,i	ō,o	ű,u
Đ,D	Ī,I	Ō,O	Ű,U	ċ,c	ĩ,i	ŏ,o	ŵ,w
Ď,D	Ĭ,I	Ŏ,O	Ų,U	č,c	ī,i	ő,o	ý,y
È,E	Į,I	Ő,O	Ŵ,W	ð,d	ĭ,i	ŕ,r	ÿ,y
É,E	İ,I	Ŕ,R	Ý,Y	è,e	į,i	ŗ,r	ŷ,y
Ê,E	Ķ,K	Ŗ,R	Ÿ,Y	é,e	ķ,k	ř,r	ž,z
Ë,E	ĸ,K	Ř,R	Ŷ,Y	ê,e	ĺ,l	š,s	ź,z
Ē,E	Ĺ,L	Š,S	Ž,Z	ë,e	ļ,l	ś,s	ż,z

Example

This example shows the input data and how the diacritical characters are replaced on output.

Table 172: Diacritical character removal example

Input	Output
María Hernández	Maria Hernandez
Geschäftsführer	Geschaeftsfuehrer
Telecomunicações São Paulo	Telecomunicacoes Sao Paulo

Related Information

Other standardization options [page 1157]

3.5.5 Platform transforms

3.5.5.1 Case

Specifies multiple paths in a single transform (different rows are processed in different ways).

The Case transform simplifies branch logic in data flows by consolidating case or decision making logic in one transform. Paths are defined in an expression table.

3.5.5.1.1 Data inputs

Only one data flow source is allowed.

3.5.5.1.2 Editor

The Case transform editor consists of a Schema In pane and a Case tab.

The Case tab includes:

- An expression table that lists labels and their CASE expressions
- An embedded Smart Editor for the CASE expression
- A *Functions* button that open the Function Wizard
- An ellipses (...) button that opens a full-size Smart Editor

1. Use the buttons, or right-click the expression table to insert or delete cases.

   While using this table, the window also allows you to:

   - Select multiple rows
   - Apply delete functionality to a multiple selection
   - Press Delete or Insert keys to delete or add an expression
2. In the expression table, click a label to rename it.
3. Enter the expression in the editor (drag columns from the input schema).

   For large expressions, open the smart editor. Both the smart editor and the function wizard can assist you with expression creation.
4. To add a DEFAULT case, select the *Produce default output when all expressions are false* check box. The label changes to read *Produce default output with label: default*. Change the label name if desired.

   When you add a DEFAULT case, Data Services will send rows to this case when all other case conditions are false.

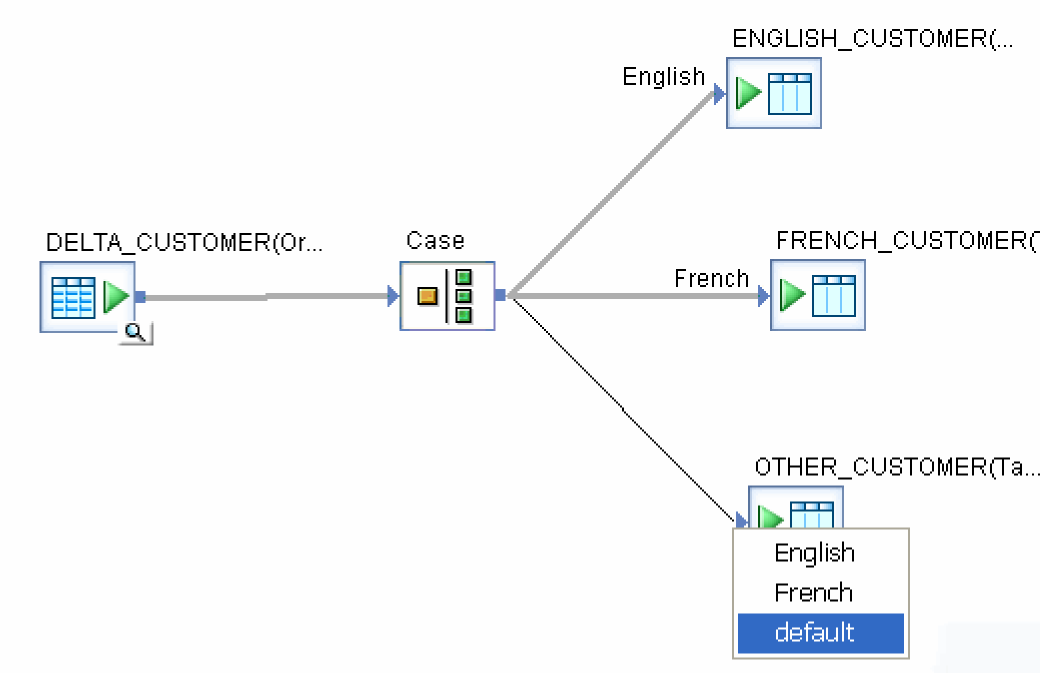
3.5.5.1.3 Options

Option	Description
Expression	CASE expression for the corresponding label.

Option	Description
	DEFAULT is the expression used when all other CASE expressions evaluate to false. To enable DEFAULT, select the *Produce default output when all expressions are false* check box. The label changes to read *Produce default output with label: default*. Change the label name if desired.
Label	Name of the connection description indicating where data will go if the corresponding CASE condition is true.
Preserve expression order	This option is available only when the *Row can be TRUE for one case only* option is checked. Select this option if expression order is important to you. When this option is NOT checked, you can increase the performance of the Case transform, because Data Services will reorder your expressions to process them in a less CPU intensive manner first. This reordering of expressions can change your results, because there is no way to guarantee which expression will evaluate to TRUE first.
Produce default output with label or *Produce default output when all expressions are false*	This option changes depending on whether it is selected. Select *Produce default output with label* to send rows to this case when all other case conditions are false.
Row can be TRUE for one case only	If this option is selected, the row is passed to the first case whose expression returns TRUE. Otherwise, the row is passed to all the cases whose expression returns TRUE. For jobs created in the 6.0 release or earlier, this option is set to FALSE. When you create a case transform in 6.1or later, this option defaults to TRUE

3.5.5.1.4 Data outputs

Connect the output of the Case transform with another object in the workspace. Choose a case label from a pop-up menu. Each label represents a case expression (WHERE clause) created in the Case editor.



The connections between the Case transform and objects used for a particular case must be labeled. Each output label in the Case transform must be used at least once.

To delete a case connection, right-click the connection label and select *Delete*.

The Case transform can be used to implement IF-THEN-ELSE logic rather than using a conditional flow. However:

- Conditionals operate at the work flow level
- Case transforms operate within data flows

3.5.5.2 Map_Operation

Modifies data based on mapping expressions and current operation codes. The operation codes can be converted between data manipulation operations.

> Note
>
> If you configure both mapping expressions and operation codes, Data Services runs mapping expressions before any row type conversions. In addition, mapping expressions are optional, so if you don't have any configured Data Services performs operation codes only.

Writing map expressions per column and per row type (INSERT/UPDATE/DELETE) allows you to:

- change the value of data for a column.
- execute different expressions on a column, based on its input row type.
- use the before_image function to access the before image value of an UPDATE row.

> Note
>
> You can use mapping expressions on only top-level schemas. They do not work on nested schemas.

This transform can also change operation codes on data sets to produce the desired output. For example, if a row in the input data set has been updated in some previous operation in the data flow, you can use this transform to map the UPDATE operation to an INSERT. The result of converting UPDATE rows into INSERT rows is the preservation the rows in the target.

Data Services can push Map_Operation transforms to the source database.

Related Information

Effective_Date [page 1081]
real [page 1036]
before_image [page 1528]

3.5.5.2.1 Data inputs

A data set with rows flagged with any operation codes.

The input data set can contain hierarchical data.

Use caution when using columns of data type `real` in this transform. Comparison results are unpredictable for this data type.

3.5.5.2.2 Options

Tab	Option	Description
Mapping	*Update mapping* *Insert/Normal mapping* *Delete mapping*	Allows you to enter and edit optional mapping expressions for output columns and row types.
Map Operations	*Output row type*	Indicate the new operations desired for the input data set. Choose from the following operation codes: INSERT, UPDATE, DELETE, NORMAL, or DISCARD.

3.5.5.2.3 Data outputs

A data set with rows flagged as specified by the mapping operations.

Rows in the input data set can contain any of the following operation codes:

- NORMAL
- INSERT
- DELETE
- UPDATE

Any of these operation codes can be mapped to:

- NORMAL
- INSERT
- DELETE
- UPDATE

In addition, the DISCARD option can be assigned. Discarded rows are not passed through to the output of the transform.

By default, every input operation type maps to itself. For each specified mapping, every row in the input data set that matches the input mapping operation is converted to the specified output operation.

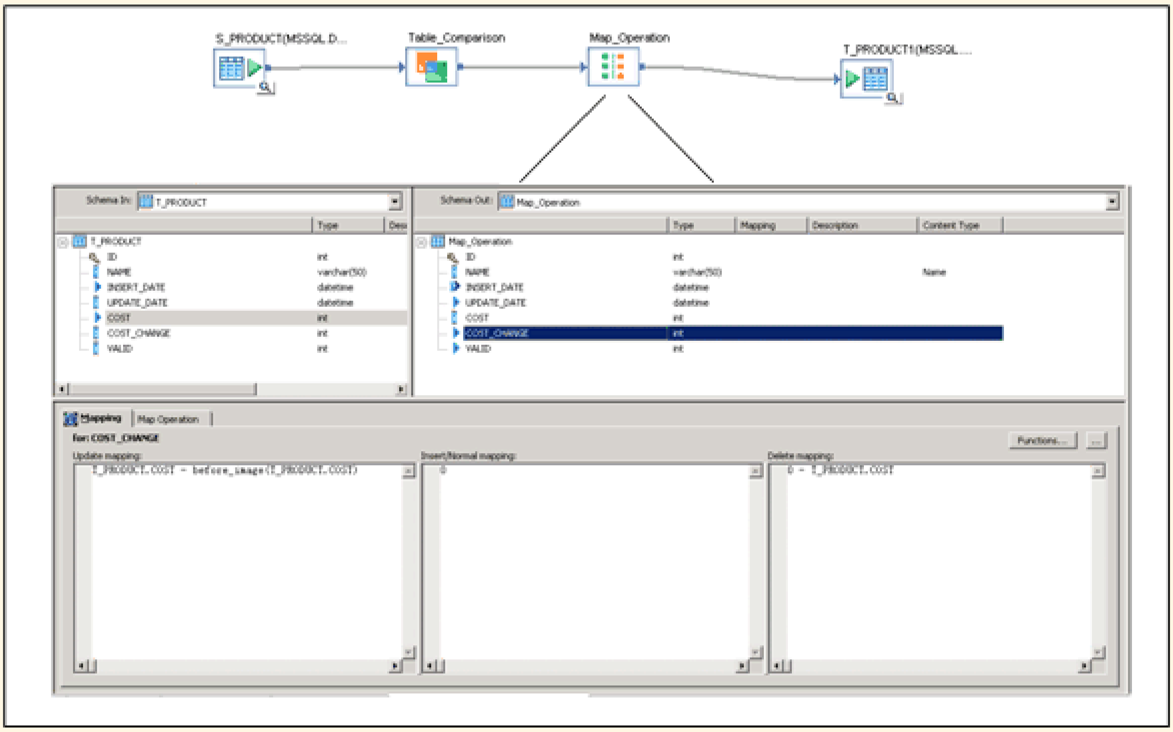
Example

Showing a change in cost

In this example, the original target table shows the cost field value as 9,000 and the cost_change field value as 0. Later on, the input data changes so that the cost field value is 1,000.

To see what the change in cost is after the input changes, you would do the following:

1. Set up the Table_Comparison transform to determine the row type (for example, INSERT or UPDATE).
2. In the Map_Operation transform, the Mapping tab would contain an expression for each column and each row type.
    - For the INSERT row type, fill the COST_CHANGE field with 0.
    - For the UPDATE row type, the before image value of the COST field (before_image(T_PRODUCT1.COST)) would be subtracted from the updated value of the COST field (T_PRODCUT1.COST). This gives you a COST_CHANGE field value of negative 8000 (1,000 - 9,000 = -8,000).
    - For the DELETE row type, the COST_CHANGE field is set as 0 - T_PRODUCT1.COST.



Schema In: T_PRODUCT

	Type
T_PRODUCT	
ID	int
NAME	varchar(50)
INSERT_DATE	datetime
UPDATE_DATE	datetime
COST	int
COST_CHANGE	int
VALID	int

Schema Out: Map_Operation

	Type	Mapping	Description	Content Type
Map_Operation				
ID	int			
NAME	varchar(50)			Name
INSERT_DATE	datetime			
UPDATE_DATE	datetime			
COST	int			
COST_CHANGE	int			
VALID	int			

Mapping | Map Operation

For: COST_CHANGE

Update mapping:

```
T_PRODUCT.COST - before_image(T_PRODUCT.COST)
```

Insert/Normal mapping:

```
0
```

Delete mapping:

```
0 - T_PRODUCT.COST
```

3. The Map Operation tab would have the following settings:

Input row type	Output row type
normal	normal
update	update
insert	insert
delete	update

Example

Changing a field value and a row type

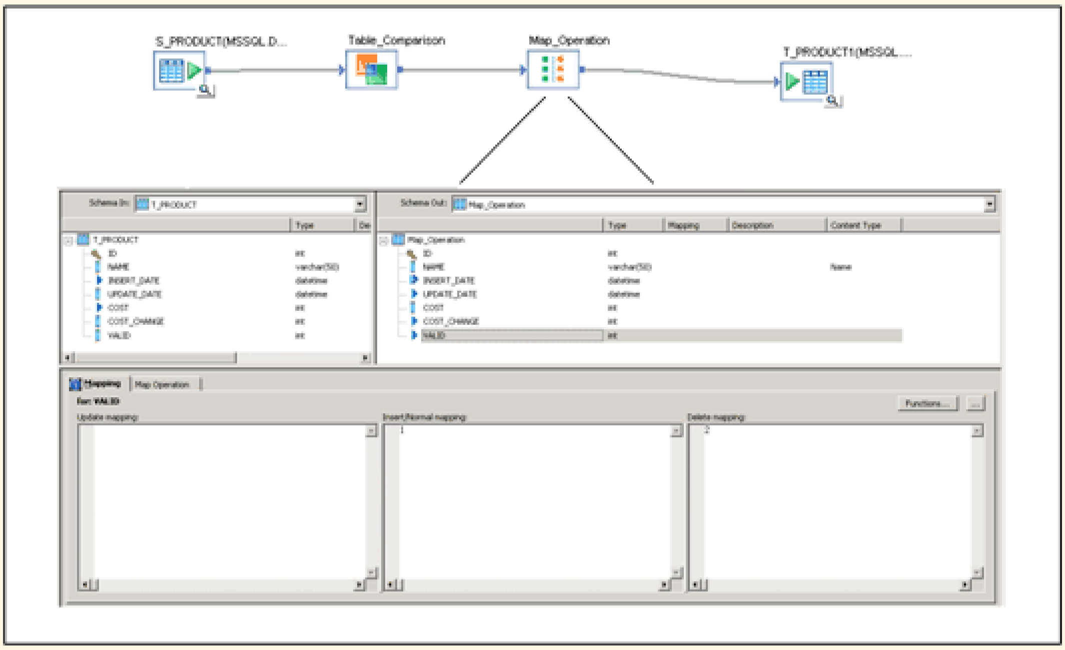
In this example, an expression is created to set the VALID field to 0 and to convert the DELETE row type to an UPDATE row type.

The VALID field (1=valid, 0=invalid) indicates if a product is in production or not. You do not want to delete the product from the target even though it has been deleted from the source. Instead, you want to update the valid field in the target to 0.

Note

When configuring both mapping expressions and operation codes, Data Services runs mapping expressions before any row type conversions.

The Mapping tab would appear as follows:



Schema In: T_PRODUCT

T_PRODUCT	Type
ID	int
NAME	varchar(50)
INSERT_DATE	datetime
UPDATE_DATE	datetime
COST	int
COST_CHANGE	int
VALID	int

Schema Out: Map_Operation

Map_Operation	Type	Mapping	Description	Content Type
ID	int			
NAME	varchar(50)			Name
INSERT_DATE	datetime			
UPDATE_DATE	datetime			
COST	int			
COST_CHANGE	int			
VALID	int			

For: VALID

Update mapping:

Insert/Normal mapping: 1

Delete mapping: 2

The Map Operation tab would have the following settings:

Input row type	Output row type
normal	normal
update	update
insert	insert
delete	update

Related Information

before_image [page 1528]

Table_Comparison [page 1110]

3.5.5.3 Merge

Combines incoming data sets, producing a single output data set with the same schema as the input data sets.

3.5.5.3.1 Data inputs

A data set from two or more sources with rows flagged as any operation code.

All sources must have the same schema, including:

- The same number of columns
- The same column names
- The same data types of columns

If the input data set contains hierarchical data, the names and data types must match at every level of the hierarchy.

3.5.5.3.2 Options

None.

3.5.5.3.3 Data outputs

A data set consisting of rows from all sources, with any operation codes. The output data has the same schema as the source data, including nested schemas.

The output data set contains a row for every row in the source data sets. The transform does not strip out duplicate rows. If columns in the input set contain nested schemas, the nested data is passed through without change.

If the data types of columns in the sources do not match the target, add a query in the data flow before the Merge transform. In the query, apply a data type conversion to the columns with data types that do not match the target column data types.

You must apply other operations such as DISTINCT in a query following the Merge transform.

3.5.5.4 Query

The Query transform retrieves a data set that satisfies conditions that you specify. A Query transform is similar to a SQL SELECT statement.

Related Information

Query transform output schema [page 1445]
Query transform input schema [page 1443]
Smart editor [page 1021]

3.5.5.4.1 Data inputs

A query has data inputs, which are data sets from one or more sources with rows flagged as any operation code.

> i Note
>
> Use caution when using columns of data type `real` in this transform. Comparison results are unpredictable for the `real` data type.

3.5.5.4.2 Data outputs

The Query transform retrieves a data set that satisfies conditions that you specify. A Query transform is similar to a SQL SELECT statement.

A query has data outputs, which are data sets based on the conditions that you specify using the schema specified in the output schema area.

3.5.5.4.3 Editor

Use the Query editor to specify the Schema In, Schema Out, and Options for the Query transform.

The areas can be resized in order to expand the area in which you are working. You can also expand and contract the columns to change the width of properties displayed in the input and output schema areas.

3.5.5.4.4 To search in an input or output schema

1. In the Query editor Find tab, enter the search term in the *Find what* box or select from previous search terms in the drop-down list.
2. In the *Schemas* list, choose the schemas in which to search.
3. In the *Elements* list, choose the types of mappings in which to search.
4. In the *Where* list, choose the properties to search within.

> **Note**
>
> You can search within one or all properties, but not within two or three specific properties at a time.

5. Select the *Match case* check box to constrain your search to the capitalization entered.
6. Click *Find*.
   The Designer searches the query for the words you specified within the constraints you defined.

> **Note**
>
> The Designer searches for columns loaded into memory. If columns are not loaded into memory, you must expand the schema to load the columns into memory before clicking *Find* and searching for the columns.

All matches are shown in the box below the find constraints. When you click to select a table or column name, the table or column is automatically highlighted and shown in the corresponding input or output schema area.

Initially, the Designer lists the matching columns in the order that they appear within the schemas. If you are searching both schemas, the Designer lists the first match found in the input schema first and the last match found in the output schema last. You can sort the list of matches by property. Each time you click a property heading, the Designer resorts the matches, cycling through original order, ascending order, and descending order.

Arrow icons confirm column and sort type. For example, if you sort the data by the *Description* property and in ascending order, an "up" arrow appears next to the *Description* heading. Click the heading again and a "down" arrow appears to indicate that the data is now sorted in descending alpha-numeric order. Click again and the match list returns to its original sort order.

3.5.5.4.5 Query transform input schema

The input schema area displays all schemas input to the Query transform as a hierarchical tree. Each input schema can contain zero or more of the following elements:

- Columns
- Nested schemas

Icons preceding columns are combinations of the following graphics:

Icon	Description
🔑	Primary key.
▯	Column that is not used in output mapping.
▶	Column that is used in output mappings.

The *Input* list at the top left of the query editor indicates the schema that is currently selected. As you select schemas or columns in the input schema area, the *Input* list displays the corresponding schema. Conversely, you can select a schema in the *Input* list to move easily to a required schema.

You can right-click elements in the input schema area and select the following menu commands:

Command	Applicable elements	Effect
Copy	Columns, schemas	Stores a copy of the selected elements in the clipboard, leaving the elements in the input schema area.
Find	Anywhere in input schema area	Locates an output element with the name or description you enter.
Refresh	Anywhere in input schema area	Refreshes the display of the input schema area.
Parent	Columns	Selects the parent schema of the selected column.
Collapse	Columns, schemas	Collapses a selected schema or a selected column's parent schema (to facilitate viewing/navigation).
Generate DTD...		Generates a DTD format that corresponds to the structure of the selected schema (either NRDM or relational). Generates all data types as varchar.
Generate XML Schema...		Generates an XML Schema that corresponds to the structure of the selected schema (either NRDM or relational). All data types match those of the selected schema.
Propagate Column From...	Columns	Carries a selected column schema from an upstream source or transform through intermediate objects to the input schema. Simple mappings are created in each object with no change to the data type or data itself.
Map to Output	Columns, schemas	Creates a simple mapping from the input schema area to the output schema area.
Create File Format...		Creates a file format from a relational table schema. All data types match those of the original table schema.
Properties	Columns, schemas	Displays the properties of the selected element. You cannot modify the properties.

3.5.5.4.6 Query transform output schema

The output schema area displays the schema output from the Query transform as a hierarchical tree. The output schema can contain one or more of the following elements:

- Columns
- Nested schemas
- Functions

Icons preceding columns are combinations of the following graphics:

Icon	Description
🔑	Primary key.
▶	Column that has a simple mapping. A simple mapping is either a single column or an expression with no input column (that is, an expression that does not vary with input).
⏩	Column that has a complex mapping. A complex mapping is any mapping that is not simple.
✗	(Red cross superimposed on any icon) Incorrect mapping. Data Services does not perform a complete validation during design, so the editor may not flag an incorrect mapping. For a complete validation, select ▶ *Validation* > *Validate* ◀.

The Schema Out pane shows the following:

- The current schema in the Schema Out list at the top and in the output schema area. The current schema determines:
  - ○ The output elements that you can modify (add, map, or delete).
  - ○ The scope of the SELECT through ORDER BY tabs in the options area.
- Non-current schemas appear dim in the output schema area.

Related Information

Change the current schema [page 1445]

3.5.5.4.7 Change the current schema

There are several ways to change the current schema in a Query transform:

- Select a schema from the Output list.
- Right-click a schema, column, or function in the output schema area and select *Make Current*.
- Double-click one of the non-current (dim) elements in the output schema area.

When you connect a target table to a query with an empty output schema, SAP Data Services automatically fills the query's output schema with the columns from the target table, without mappings.

The software only fills the target schema in the output of a query when you connect a target table to a query with an empty output schema. If the output schema contains any column mappings, the software does not overwrite those mappings. Similarly, if you connect a query to one target, and then disconnect that target and connect to another target, the output schema will show the columns from the first target connected.

The software does not fill the output schema from a file, an XML message, an IDoc, or any other target.

There are several techniques to change the output schema:

- Drag and drop (or copy and paste) columns or nested schemas from the input schema area to the output schema area (this provides simple column mappings).
  If you drop a column on an existing column, you can remap that column. Select Remap Column to update only the column mapping or select Remap with Data Type to update the column mapping and data type. Alternatively, you can select Insert Above or Insert Below to add the column as a new mapping or Cancel if you do not want to add the column to the output schema.
- Right-click the current schema and select *New Output Column* or *New Output Schema*. You can provide simple column mappings by dragging input columns over the new output columns. For complex mappings, use the options area.
- Right-click the current schema and select *New Function*. The function must already be imported into the repository. You can add adapter functions and SAP RFC or BAPI functions. These functions return multiple columns (in contrast to the functions used in mappings and Where clauses, which return single values). In the Define Input Parameter(s) window, map all first-level input parameters for the function to the input parameters of the query.
- Right-click columns in the current schema to assign and reverse primary key settings on output columns. A key icon indicates primary keys.
- Right-click the current schema and select *Unnest* to flatten output schemas. Use this command when a job has a source with a nested schema (such as an XML file), and you map columns from this source to a flat target table schema.

You can right-click elements in the output schema area and select commands. Generally, the elements must be within the current schema.

Command	Applicable elements	Effect
Cut	All	Removes the selected elements from the output schema area and stores a copy of the elements in the clipboard.
Copy	All	Stores a copy of the selected elements in the clipboard, leaving the elements in the output schema area.
Paste	All	Inserts the elements stored in the clipboard at the current cursor location (this must be within the current schema). Only visible when the clipboard contains something. If the cursor overlaps an existing column, you are prompted to insert above, insert below, remap column, or cancel.

Command	Applicable elements	Effect
		Note Copying an input element and pasting it in the output schema area provides a simple mapping from the input element to the output element. You can also do this by dragging the input element to the output schema area.
Delete	All	Removes the selected elements from the output schema area (without making a copy).
Find...	All	Locates an output element with the name or description you enter.
Make Current	All outside the current schema	Makes the selected schema, or the schema of the selected element, the current schema.
New Output Column...	Schemas	Adds an output column to the current schema with the name and properties you enter.
New Output Schema...	Schemas	Adds a nested schema to the current schema with the name you enter.
New Function Call	Schemas	Adds a function or stored procedure call to the current schema. The function or procedure must already be imported into the repository. You can add adapter functions and SAP RFC or BAPI functions. These functions return multiple columns (in contrast to the functions used in mappings and Where clauses that return single values). In the Define Input Parameter(s) window, map all first-level input parameters for the function to the input parameters of the query.
Modify Function Call	Functions	Allows you to modify the selected function. In the Define Input Parameter(s) window, map all first-level input parameters for the function to the input parameters of the query.
Propagate Column From	Columns	Carries a selected column schema from an upstream source or transform through intermediate objects to the output schema. Simple mappings are created in each object with no change to the data type or data itself.
Unnest	Nested schemas	Toggles to flatten or re-nest the selected schema into the parent schema in the query output. An unnested schema will not appear in the succeeding

Command	Applicable elements	Effect
		transform or target; only its columns appear. Unnested schemas appear in the Query transform output schema area as table icons with a black arrow that points to the left.
Nest with sub-schemas	Nested schemas	Re-nests the selected schema and all sub-schemas into the parent schema in the query output.
Unnest with sub-schemas	Nested schemas	Flattens the selected schema and all sub-schemas into the parent schema in the query output. Unnested schemas and sub-schemas do not appear in the succeeding transform or target; only the columns appear. Unnested schemas and sub-schemas appear in the Query transform output schema area as table icons with a black arrow that points to the left.
Primary Key	Columns	Toggles the primary key attribute of the column on (check mark appears next to the command) or off (no check mark appears next to the command). A key icon indicates that a column is a primary key.
Optional	Schemas	Toggles to make a schema optional.
Generate DTD		Generates a DTD format that corresponds to the structure of the selected schema (either NRDM or relational). Generates all data types as varchar.
Generate XML Schema		Generates an XML Schema that corresponds to the structure of the selected schema (either NRDM or relational). All data types match those of the selected schema.
Create File Format		Creates a file format from a relational table schema. All data types match those of the original table schema.
Properties	All	Displays the properties of the selected element.

Related Information

Query transform output schema [page 1445]

3.5.5.4.8 Options

The options area of the Query Editor contains several tabs where you enter information to specify the data you want retrieved. Specifying information on these tabs is similar to specifying a SQL SELECT statement. Tabs containing entries are flagged by a special ℹ icon.

When you drag and drop (or copy and paste) input columns to the output schema, Data Services inserts values in the Mapping and FROM tabs. For simple mappings, this is often sufficient.

For more complex mappings, complete the appropriate tabs.

Table 173: Query Editor tab descriptions

Tab	Description
Mapping	Specifies how the selected output column will be derived (or mapped).
SELECT	Specifies distinct rows to output (discarding any identical duplicate rows).
FROM	Lists all input schemas. Allows you to specify join pairs and join conditions as well as enter join rank and cache for each input schema. The resulting SQL FROM clause is displayed.

WHERE	Specifies conditions that determine which rows are output.
	Enter the conditions in SQL syntax, like a WHERE clause in a SQL SELECT statement. For example:
	`TABLE1.EMPNO = TABLE2.EMPNO`
	`AND TABLE1.EMPNO > 1000`
	`OR TABLE2.EMPNO < 9000`
	Use the *Functions*, *Domains*, and *smart editor* buttons for help building expressions.

GROUP BY	Specifies how the output rows are combined (if required).
ORDER BY	Specifies how the output rows are sorted (if required).
Advanced	Creates separate sub data flows to process resource-intensive query clauses.
Find	Enables you to search for a specific word or term in the input and/or output schemas.

Note

Use the SELECT through ORDER BY tabs to specify additional constraints for the current schema, similar to SQL SELECT statement clauses.

Related Information

Mapping tab [page 1450]

SELECT tab [page 1451]

FROM tab [page 1451]

WHERE tab [page 1452]

GROUP BY tab [page 1453]

ORDER BY tab [page 1454]
Advanced tab [page 1454]
Find tab [page 1455]

3.5.5.4.9 Query Editor tabs

3.5.5.4.9.1 Mapping tab

Use the Mapping tab to specify how the selected output column is derived (or mapped). You can specify any valid expression.

Most commonly, mapping expressions contain table columns and functions.

- Enter input column names or drag columns from the input schema and drop them in the box on the Mapping tab.
- Insert functions by entering them directly, using the smart editor, or by clicking the Functions button to open the function wizard.

> Note
>
> You cannot add comments to a mapping clause in a Query transform. For example, the following syntax is not supported in the Mapping tab:

```
table.column # comment
```

If you add comments, the job will not run and you cannot successfully export it. Use the object description or workspace annotation feature instead.

After you map your source to the Query transform, you might determine that you need to use another transform before you send the data to the Query transform. For example, you might add a validation transform to ensure that only data with a certain format is passed or you might add a Case transform to send only a subset of the data.

In general, when you change an input schema to the Query transform, the Designer checks the existing top-level mappings to determine if any remapping is required.

- If the mapping contains a column with a table name that is not a current input schema name and the column is in the new input schema, the Designer automatically replaces the table name with the new input schema name. Specifically, the Designer automatically updates the input schema name for each matching column in the following option tabs of the Query editor:
  - Mapping
  - FROM
  - WHERE
  - ORDER BY
  - GROUP BY
- If the mapping contains a column that was in the obsolete table, but the column does not exist in the new input schema, you must either remove the column or remap it from the original source.

The Designer does not automatically remap the input schema for the following situations:

- When you connect a new source to the Query transform before you disconnect the old source. You must click the *Schema Remapping* button on the Mapping tab to update the input schema name for columns in the Mapping, WHERE, GROUP BY, and ORDER BY tabs.
- When the source is a nested schema and you either change the source to a similar nested schema, or you add or delete a transform before the Query transform. Click the *Schema Remapping* button to update the Mapping input schema name.

3.5.5.4.9.1.1 To remap when automatic remapping was not done in the Query transform

1. In the Mapping tab, click the *Schema Remapping* button. The *Replace Obsolete Schema window* opens.
2. In the *Specify obsolete schema* drop-down list, choose the source schema that you disconnected from the Query transform.
   This list displays only the top-level input schema. For an obsolete nested schema, enter the name of the top-level schema.
3. In the *Choose correct schema* list, choose the output schema of the transform that you added between the source and Query transform.
4. Click *Remap*. A message displays the number of columns that were remapped; for example:

```
Schema "ODS_SALESORDER" was replaced by schema "Validation_Pass" in 11 column
names.
```

3.5.5.4.9.2 SELECT tab

Use the SELECT tab to output only distinct rows (discarding any identical duplicate rows).

To discard duplicate rows, select the *Distinct rows* check box. This is similar to specifying a SELECT DISTINCT SQL statement.

3.5.5.4.9.3 FROM tab

Use the FROM tab to specify input schemas as well as join information, such as join pairs, join type, and the join condition used in the current output schema. The specified information is similar to the FROM clause in a SQL SELECT statement.

The *FROM* tab is divided into three areas:

- The uppermost area contains information about the source tables connected to the Query transform in the data flow. The Input schemas column is populated with the names of the source tables.
- The middle area, *Join pairs*, allows you to specify the tables to be joined, the type of join, and the join condition.
- The lower area, *FROM clause*, displays the SQL FROM clause and is automatically updated as you add join conditions. This area is read-only, but can be copied to the clipboard.

The following table describes the columns displayed in the *Join pairs* area.

Column name	Description
Left	The left source of a join. For the first join pair, select the input schema from the drop-down list of available schemas. For subsequent join pairs, the result of the previous join pair is taken as the left source and the schema cannot be specified.
Join Type	The type of join. Valid values are Inner join and Left outer join. If a table is not explicitly joined to another table, then it is cross-joined (Cartesian Product) to the result of the final join defined by the join pairs.
Right	The right source of a join. Any input schema not used in a previous join pair.
smart editor ...	Optional: click the icon to open the smart editor. Within the smart editor *Data* tab, you can drag and drop columns to specify the join condition.
Propose Join ◉	Optional: Click *Propose Join* to have Data Services generate a join expression. The SQL clause is automatically updated after you change the left source, right source, or join type.
Join condition	A join condition is required for each join pair. Where possible, Designer automatically suggests a join condition based on the input schemas of the join pair. To edit the join condition, you can enter the join condition field or use the smart editor.

i Note

If your expression contains varchar comparisons, Data Services ignores trailing blanks in the data. For Oracle data, use the rtrim or rpad functions if the number of trailing blanks might differ on either side of the comparison.

3.5.5.4.9.4 WHERE tab

Use the WHERE tab to set conditions that determine which rows are output. Enter the conditions in SQL syntax, as you would a WHERE clause in a SQL SELECT statement. The *Propose* button generates possible join conditions. You can specify the following conditions:

Condition	Example
Data set filters	`TABLE1.EMPNO > 1000`

Multiple conditions using logical operators	`TABLE1.EMPNO > 1000`
	`OR TABLE2.EMPNO < 9000`

Join conditions for inner joins only Note Although it is technically possible to specify inner join conditions in the WHERE tab, the best practice is to specify inner join conditions in the FROM tab.	`TABLE1.EMPNO = TABLE2.EMPNO`

You can specify any valid expression. To enter conditions, do one of the following:

- Enter expressions in the editor.
- Drag columns from the input schema area to the editor.
- Use the Functions button. Use the pushdown_sql function to have Data Services create WHERE clauses dynamically based on data rather than pre-specifying the clause.

Note

If your expression contains varchar comparisons, Data Services ignores trailing blanks in the data. For Oracle data, use the rtrim or rpad functions if the number of trailing blanks might differ on either side of the comparison.

3.5.5.4.9.5 GROUP BY tab

Use the GROUP BY tab to specify a list of columns for which you want to combine output. For each unique set of values in the group by list, Data Services combines or aggregates the values in the remaining columns. For example, you might want to group sales order records by order date to find the total sales ordered on a particular date.

To add a column to the Group By list, select the column in the input schema area and drag it to the box in the *GROUP BY* tab. The Designer adds the column to the bottom of the list.

The first column listed is used for primary grouping, the second column listed is used for secondary grouping, and so forth. To change the groupings, use one of the following options:

- Right-click the column and select *Move Up* or *Move Down*.
- Select the column and click the down or up arrow in the top right corner of the GROUP BY tab.

To remove a column, use one of the following options:

- Right-click the column and select *Delete*.
- Select the column and click the delete icon in the top right corner of the GROUP BY tab.

To group by complex expressions (instead of by specific column values), use another query to produce a single column containing the grouping expression. Insert the new query immediately before this transform in your data flow, and specify the created column on the GROUP BY tab.

If you specify a group by list, then all columns in the output schema must be either in the group by list or mapped to an aggregate function, such as avg, count, max, min, or sum.

This tab is similar to the GROUP BY clause in a SQL SELECT statement.

3.5.5.4.9.6 ORDER BY tab

Use the ORDER BY tab to specify the columns you want used to sort the output data set. To add a column, select the column in the input schema area and drag it to the box on the ORDER BY tab. The Designer adds the column to the bottom of the list.

The first column listed is used for primary sorting, the second column listed is used for secondary sorting, and so forth. To change the column order, use one of the following options:

- Right-click the column and select *Move Up* or *Move Down*.
- Select the column and click the down or up arrow in the top right corner of the *ORDER BY* tab.

To remove a column, use one of the following options:

- Right-click the column and select *Delete*.
- Select the column and click the delete icon in the top right corner of the *ORDER BY* tab.

The default sort order is ascending. To change the order, select *Ascending* or *Descending* from the adjacent drop down box.

3.5.5.4.9.7 Advanced tab

Use the Advanced tab to set up Data Services so that it creates separate sub data flows. Sub data flows process any of the following resource-intensive query clauses:

- DISTINCT
- GROUP BY
- JOIN
- ORDER BY

Related Information

Smart editor [page 1021]
Functions and Procedures [page 1511]
pushdown_sql [page 1640]

3.5.5.4.9.8 Find tab

Use the Find tab to search for a specific word or term in the input schema or the output schema.

Related Information

To search in an input or output schema [page 1442]

3.5.5.4.10 Joins in the Query transform

You can use the Query editor to define joins involving two or more tables. Specifying information on the FROM and WHERE tabs has the effect of creating FROM and WHERE clauses in a SQL SELECT statement. Supported join types are inner join, left outer join, and cross-product.

Begin by specifying join pairs and join conditions in the FROM tab. As needed, restrict the result set in either the FROM or WHERE tab depending on the information that you need the query to return.

> Note
>
> The best practice is to define all joins in the FROM tab. However, inner joins can be defined in the WHERE tab using a WHERE clause.

For each pair of sources, the generated join proposal includes a join condition based on column names, foreign keys, or primary keys:

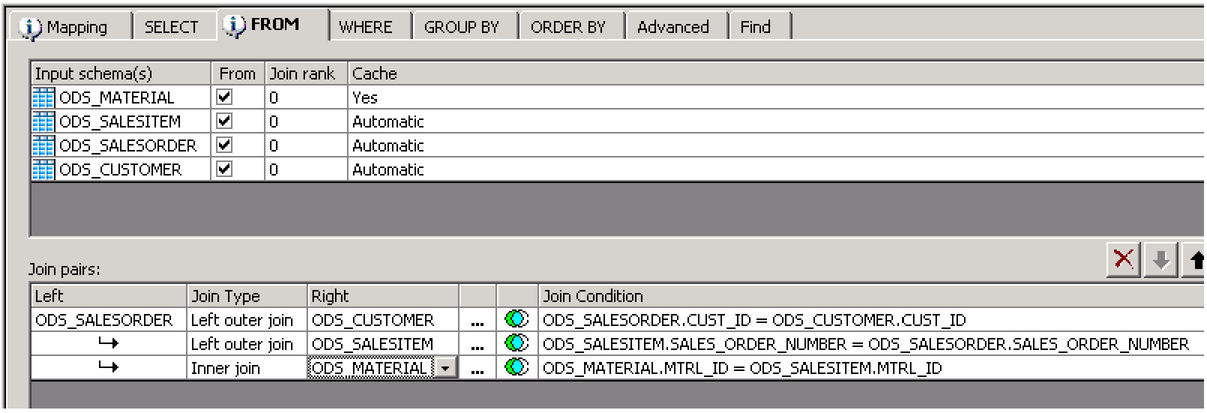
- Foreign key: If a foreign key relationship exists, Data Services adds a join condition to the expression for the columns related through keys. For example, if foreign key K2 of table T2 references primary key K1 of table T1, Data Services includes the join condition: T1.K1=T2.K2
- Primary key and column name: If a foreign key relationship does not exist, Data Services adds a join condition to the expression for columns with the same name where at least one column is part of a primary key. For example, suppose there is no foreign key relationship between tables T and S; however, both tables contain column A. Column A is part of the primary key in table S. In this example, Data Services includes the join condition: T.A = S.A

If neither foreign key nor primary key is satisfied, Data Services proposes no join condition for that pair of sources.

3.5.5.4.10.1 About join pair lists

The FROM tab allows you to create a SQL FROM clause by specifying join pair information including the tables to join, join type, and join condition. Join pairs are subject to the following requirements:

- In order to be used in a join pair, a table must be an input schema to the query.
- Inner joins and left outer joins can be specified in the same FROM clause.
- Any table from the schema list can be used as a source in at most one join pair. If a table is not explicitly joined to another table, then it will be cross-joined (Cartesian product) to the result of the final join defined by the join pairs list. A cross join (Cartesian product) is a special case of an inner join with an ON condition that always evaluates to TRUE. In other words, a cross join matches every row of one table with every row of another table.
- The join pairs list may be empty in the following cases:
  - the input schema for the query contains only one source.
  - each of the input schemas are intended to be cross-joined.
  - inner joins are defined using a WHERE clause in the WHERE tab.
- Identify a left source only for the first pair in the *Join pairs* list. All subsequent join pairs take the results of the preceding join pair as the left source.
  For example, in the following screenshot, the first join pair is a left outer join with ODS_SALESORDER as the left source and ODS_CUSTOMER as the right source. The result of that join becomes the left source of the second join pair which is a left outer join with ODS_SALESITEM as the right source. Finally the result of the two left joins becomes the left source of an inner join with ODS_MATERIAL as the right source.



Input schema(s)	From	Join rank	Cache
ODS_MATERIAL	☑	0	Yes
ODS_SALESITEM	☑	0	Automatic
ODS_SALESORDER	☑	0	Automatic
ODS_CUSTOMER	☑	0	Automatic

Join pairs:

Left	Join Type	Right			Join Condition
ODS_SALESORDER	Left outer join	ODS_CUSTOMER	...		ODS_SALESORDER.CUST_ID = ODS_CUSTOMER.CUST_ID
↳	Left outer join	ODS_SALESITEM	...		ODS_SALESITEM.SALES_ORDER_NUMBER = ODS_SALESORDER.SALES_ORDER_NUMBER
↳	Inner join	ODS_MATERIAL	...		ODS_MATERIAL.MTRL_ID = ODS_SALESITEM.MTRL_ID

- The join condition of a join pair cannot be empty and must refer to tables that are used in previous join pairs and the table used in the current join pair.
  For example, the second join pair in the screenshot above may not refer to the table ODS_MATERIAL in its join condition because ODS_MATERIAL was not used in the previous join pair. The second join pair may only refer to ODS_SALESORDER, ODS_CUSTOMER, and ODS_SALESITEM.
- If the Query transform contains only inner joins, the WHERE tab may be used to specify join conditions. However, although valid, specifying join conditions in both the FROM tab and WHERE tab is not recommended. If conditions are specified in both the WHERE and FROM tabs, they are combined to form the join conditions for the query at job execution time.
- If a query contains a left outer join, any conditions specified in the WHERE tab are treated as filters. The join conditions for a left outer join may include multiple tables and are defined in the FROM tab.
- A Query transform in an ABAP data flow cannot contain mixed inner and left outer joins. The Query transform may have only inner joins, only left outer joins, or no joins at all.

3.5.5.4.10.2 Constructing a Join Query

You can use the Query editor to specify joins involving two or more tables. The resulting SQL join types may be inner join, left outer join, or cross product.

To construct a join query:

1. Within a data flow, connect the source tables to a Query transform.
2. Click the Query transform to open the Query editor.
3. Optionally, exclude input schemas by deselecting the *FROM* checkbox in uppermost area.

   You may want to exclude an input schema if you no longer need its columns in the output schema, ORDER BY, and GROUP BY clauses.

   It also may be useful to exclude nested input schemas. However, at least two input schemas must be selected in order to create a join.
4. In the lower pane of the Query editor, click the *FROM* tab.
5. Specify the join pairs and join conditions.

   After the first join pair is specified, subsequent join pairs use the result of the previous pair as the left source. The following table describes the columns displayed in the *Join pairs* area.

Column name	Description
Left	The left source of a join. For the first join pair, select the input schema from the drop-down list of available schemas. For subsequent join pairs, the result of the previous join pair is taken as the left source and the schema cannot be specified.
Join type	The type of join. Valid values are Inner join and Left outer join. If a table is not explicitly joined to another table, then it will be cross-joined (Cartesian Product) to the result of the final join defined by the join pairs.
Right	The right source of a join. Any input schema not used in a previous join pair.
smart editor ...	Optional: click the icon to open the smart editor. Within the smart editor *Data* tab, you can drag and drop columns to specify the join condition.
Propose Join	Optional: Click *Propose Join* to have Data Services generate a join expression. The SQL clause is automatically updated after you change the left source, right source, or join type.
Join condition	A join condition is required for each join pair.

Column name	Description
	Where possible, Designer automatically suggests a join condition based on the input schemas of the join pair. To enter or edit the join condition, you can type in the join condition field or use the smart editor.

If your FROM clause contains only left outer joins or a mix of left outer joins and inner joins, you may want to change the order that the software executes the join pairs. To change the order, select a cell within a join pair row and then click the up arrow or down arrow in the upper right corner of the *Join pairs* area. Note that changing the execution order of the join pairs changes the results.

If your FROM clause contains only inner joins, changing the execution order does not change the results.

6. As necessary, to filter the result set of a left outer join, place a restriction in the ON clause or within the WHERE tab.
7. Optionally, specify the join ranks for each table in the *Join rank* column.

   The join rank indicates the rank of the source relative to other tables and files in the data flow. The join rank has no effect on the actual result produced, but can have a profound effect on join performance. The software joins sources with higher join ranks before it joins sources with lower join ranks. The order of execution depends on join rank and, for left outer joins, the order defined in the FROM clause.

   The join rank must be a non-negative integer. When set to its default value of 0, the software determines the join order. The join rank specified in the Query editor overrides any join rank specified in a source. For new jobs, specify the join rank in the Query editor.
8. As necessary, specify desired caching in the *Cache* column.

   Cache indicates whether the software should read the data from the source and load it into memory or pageable cache. Caching a source increases performance only when the data source is used as the right source of a join.

   The cache value in the Query transform takes precedence over the value specified in a source. For newly created data flows, it is preferable to specify the cache value in the Query transform.

   The default value in the Query transform is *Automatic*. Automatic assumes the value specified in the source.

   > Note
   >
   > The cache type, either in-memory or pageable, is set at the data flow level.

Related Information

Restricting left outer joins [page 1463]

Designer Guide: Nested data, Operations on nested data, Overview of nested data and the Query transform [page 356]

Performance Optimization Guide: Other tuning techniques, Join ordering, Join rank settings [page 2198]

Performance Optimization Guide: Using caches, Caching joins [page 2130]

3.5.5.4.10.3 Join Examples

3.5.5.4.10.3.1 Joins using two sources

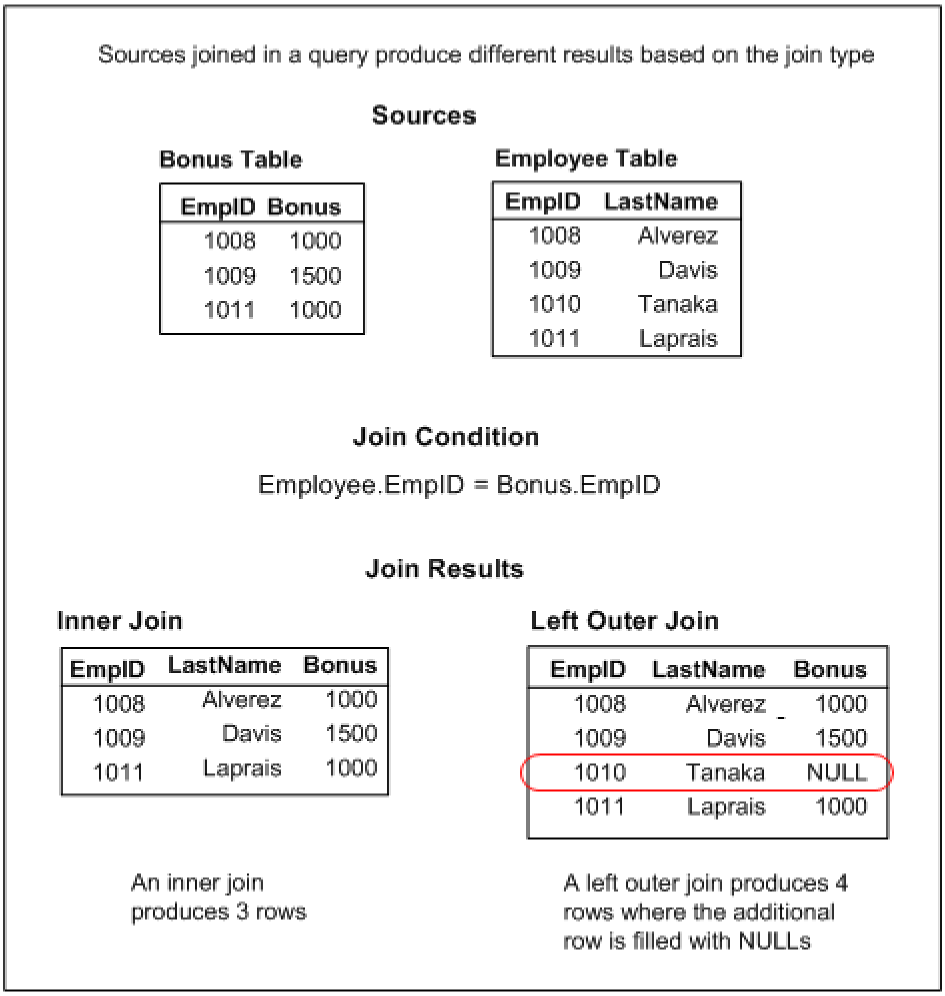
About inner and left outer joins

Sources joined in a Query transform produce different results based on the join type.

When joining two sources, an inner join returns rows from both sources where a match is found.

A left outer join returns the rows that meet the join condition, plus all of the rows in the left source that did not meet the join condition. Therefore, all rows in the left source are reproduced at least once in the result. Only data from the right source that satisfies the join condition appears in the result.

For rows from the left table that do not have corresponding data from the inner table, the missing values are assigned as null. In an inner join between the tables, the same rows would be absent in the result. The following diagram shows the difference in the join results for the sample data in the illustration:



Sources joined in a query produce different results based on the join type

Sources

Bonus Table

EmpID	Bonus
1008	1000
1009	1500
1011	1000

Employee Table

EmpID	LastName
1008	Alverez
1009	Davis
1010	Tanaka
1011	Laprais

Join Condition

Employee.EmpID = Bonus.EmpID

Join Results

Inner Join

EmpID	LastName	Bonus
1008	Alverez	1000
1009	Davis	1500
1011	Laprais	1000

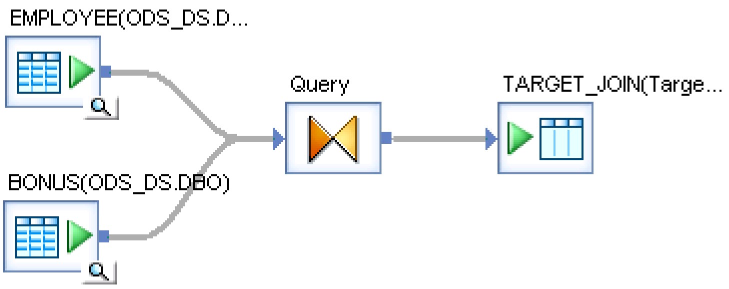
Left Outer Join

EmpID	LastName	Bonus
1008	Alverez	1000
1009	Davis	1500
1010	Tanaka	NULL
1011	Laprais	1000

An inner join produces 3 rows

A left outer join produces 4 rows where the additional row is filled with NULLs

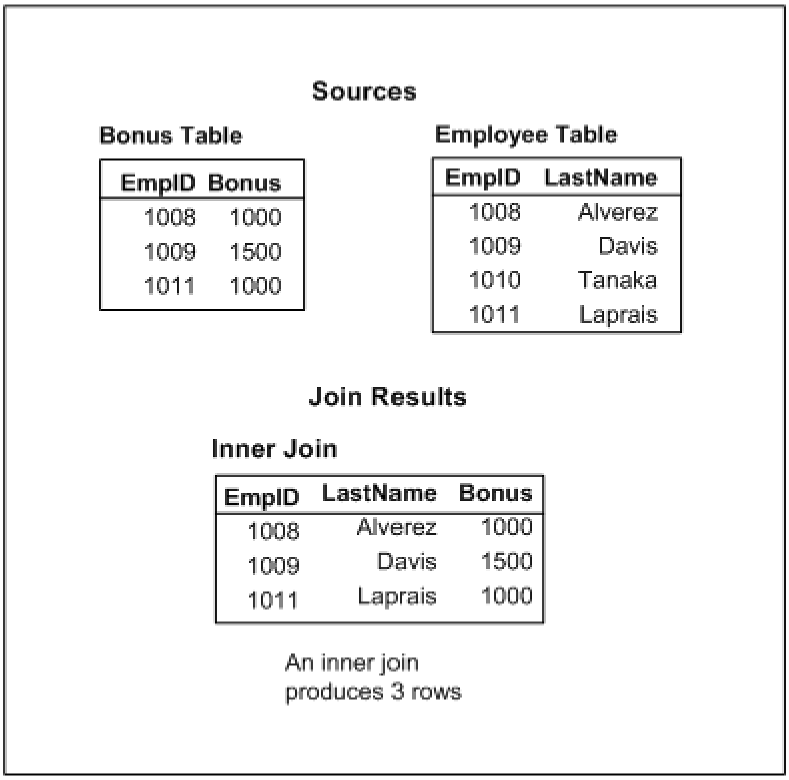
In Data Services, the two tables in the data flow would be sources to a Query transform and the join results would appear in the target table as shown below.



Inner join example

To produce a list of only the employees receiving bonuses, the data in the Employee Table and Bonus Table sources below are joined using an inner join to produce three rows where each row contains values from both sources.

The outer table is `Employee` and the inner table is `Bonus`.



Sources

Bonus Table

EmpID	Bonus
1008	1000
1009	1500
1011	1000

Employee Table

EmpID	LastName
1008	Alverez
1009	Davis
1010	Tanaka
1011	Laprais

Join Results

Inner Join

EmpID	LastName	Bonus
1008	Alverez	1000
1009	Davis	1500
1011	Laprais	1000

An inner join
produces 3 rows

The syntax of the SQL is:

```
SELECT     EMPLOYEE.EMPID, EMPLOYEE.LASTNAME,
BONUS.BONUS
FROM EMPLOYEE INNER JOIN BONUS
ON      (EMPLOYEE.EMPID = BONUS.EMPID)
```

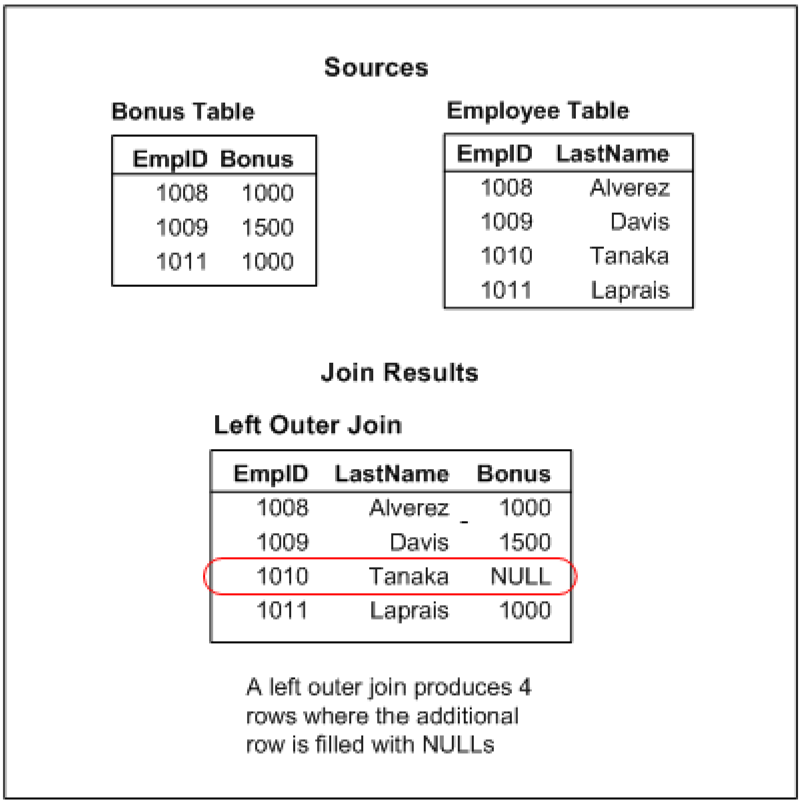
Use the Query editor *FROM* tab to define the Join pairs list as follows:

Left	`Employee`
Join Type	Inner join
Right	`Bonus`
Join Condition	`EMPLOYEE.EMPID = BONUS.EMPID`

Left outer join example

To produce a list of all employees and show which are receiving bonuses the data in the Employee Table and Bonus Table sources below are joined with a left outer join to produce four rows where the extra row contains a NULL.

The outer table is `Employee` and the inner table is `Bonus`.



Sources

Bonus Table

EmpID	Bonus
1008	1000
1009	1500
1011	1000

Employee Table

EmpID	LastName
1008	Alverez
1009	Davis
1010	Tanaka
1011	Laprais

Join Results

Left Outer Join

EmpID	LastName	Bonus
1008	Alverez	1000
1009	Davis	1500
1010	Tanaka	NULL
1011	Laprais	1000

A left outer join produces 4 rows where the additional row is filled with NULLs

The SQL query is:

```
SELECT     EMPLOYEE.EMPID, EMPLOYEE.LASTNAME,
BONUS.BONUS
```

```
FROM EMPLOYEE LEFT OUTER JOIN BONUS
ON      (EMPLOYEE.EMPID = BONUS.EMPID)
```

Use the Query editor *FROM* tab to define the Join pairs list as follows:

Left	`Employee`
Join Type	Left outer join
Right	`Bonus`
Join Condition	`EMPLOYEE.EMPID = BONUS.EMPID`

3.5.5.4.10.3.2 Mixed inner and left outer joins

When joining more than two tables in the Query editor, a left source is identified in the first pair of tables in the *Join pairs* list. All subsequent join pairs take the results of the preceding join pair as the left source.

Mixed joins example

The example below illustrates how sequential joins can produce a result showing all of the departments that have employees and the employees' bonuses. In this case, the Department table would be the left source and the Employee table would be the right source of an inner join. The results of that inner join would then be joined to the Bonus table using a left outer join The results of the inner join would be the left source and the Bonus table would be the right source.

The joins produce four rows with the bonus information NULL where there was no value in the bonus table.

Sources:

Department table		Employee table			Bonus table	
DeptID	**Department**	**EmpID**	**LastName**	**DeptID**	**EmpID**	**Bonus**
01	Accounting	1008	Alverez	01	1008	1000
02	Finance	1009	Davis	02	1009	1500
03	Sales	1010	Tanaka	01	1011	1000
04	Marketing	1011	Laprais	01		

Join Results:

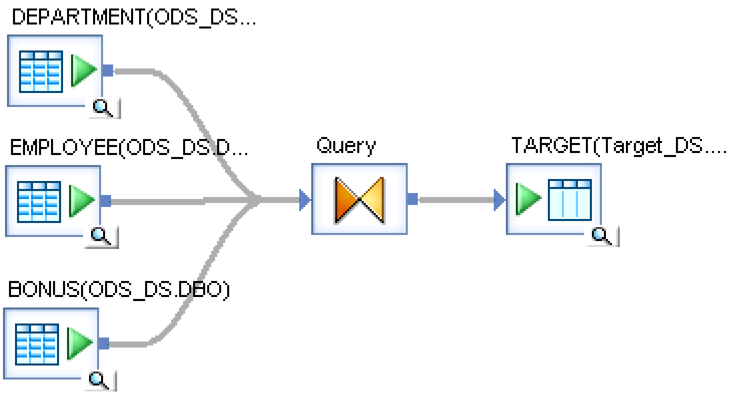
DeptID	Department	LastName	Bonus
1	Accounting	Alverez	1000
1	Accounting	Tanaka	NULL
1	Accounting	Laprais	1000

DeptID	Department	LastName	Bonus
2	Finance	Davis	1500

The SQL query is:

```
SELECT     DEPARTMENT.DEPTID, DEPARTMENT.DEPARTMENT,
  EMPLOYEE.LASTNAME, BONUS.BONUS
FROM (DEPARTMENT INNER JOIN EMPLOYEE
(ON DEPARTMENT.DEPTID=EMPLOYEE.DEPTID))
LEFT OUTER JOIN BONUS
ON      (EMPLOYEE.EMPID = BONUS.EMPID)
```

In Data Services, the three tables in the data flow would be sources to a Query transform and the join results would appear in the target table as shown below.



Use the Query editor *FROM* tab to define the Join pairs list as follows:

Left	Join Type	Right	Join Condition
DEPARTMENT	Inner join	EMPLOYEE	DEPARTMENT.DEPTID=EMPLOYEE.DEPTID
—»	Left outer join	BONUS	EMPLOYEE.EMPID=BONUS.EMPID

3.5.5.4.10.3.3 Restricting left outer joins

About restricting left outer joins

An unrestricted left outer join between two tables results in all of the rows from the left outer table along with data from the inner table that satisfies the join condition. NULL values are assigned in rows that do not contain data.

The result set for a left outer join that is restricted depends on whether the restriction is placed in the ON condition or in the WHERE clause. Where you place the restriction ultimately depends on what information you need the query to return. The following table shows the result set defined in each case and how to define the query in the Query editor.

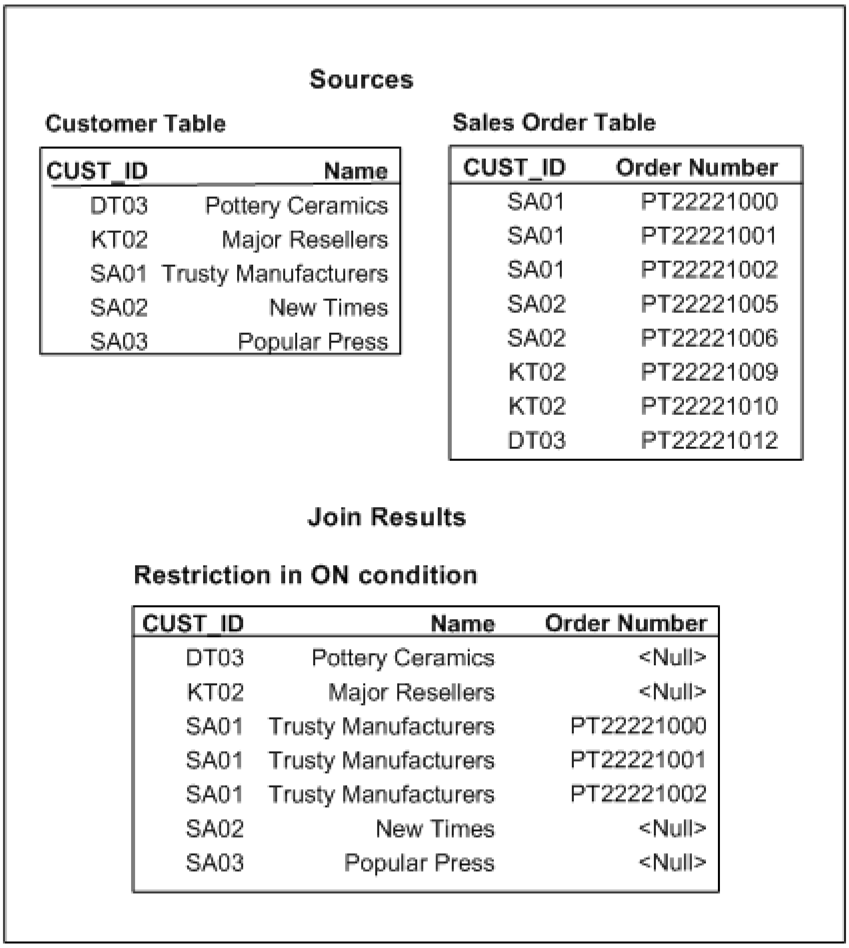
Result set	Restriction	Query editor tab to use
Joined table includes all rows of the left source (including null values).	ON	FROM
Joined table includes only rows for which the restriction is true.	WHERE	WHERE

i Note

For inner joins, it does not matter where the restriction is placed; the result set is the same in either case.

Restriction placed in the ON condition example

For example, assume CUSTOMER is the left source and SALESORDER is the right source of a left outer join. The diagram below shows data in the source tables and the results of a left outer with the restriction placed in the ON condition.



Sources

Customer Table

CUST_ID	Name
DT03	Pottery Ceramics
KT02	Major Resellers
SA01	Trusty Manufacturers
SA02	New Times
SA03	Popular Press

Sales Order Table

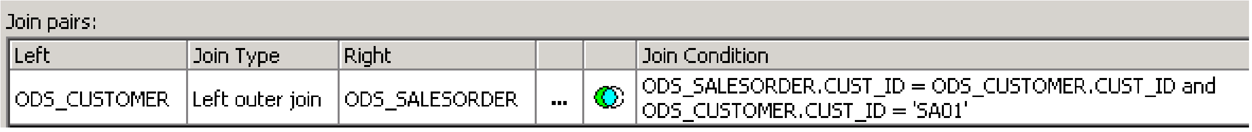
CUST_ID	Order Number
SA01	PT22221000
SA01	PT22221001
SA01	PT22221002
SA02	PT22221005
SA02	PT22221006
KT02	PT22221009
KT02	PT22221010
DT03	PT22221012

Join Results

Restriction in ON condition

CUST_ID	Name	Order Number
DT03	Pottery Ceramics	<Null>
KT02	Major Resellers	<Null>
SA01	Trusty Manufacturers	PT22221000
SA01	Trusty Manufacturers	PT22221001
SA01	Trusty Manufacturers	PT22221002
SA02	New Times	<Null>
SA03	Popular Press	<Null>

The join condition specified in the FROM clause contains the clause `AND CUSTOMER.CUST_ID = 'SA01'` and the result returns all rows of the left source, CUSTOMER, including those rows with NULL values. The Join pairs area of the FROM tab would appear as follows:



Join pairs:

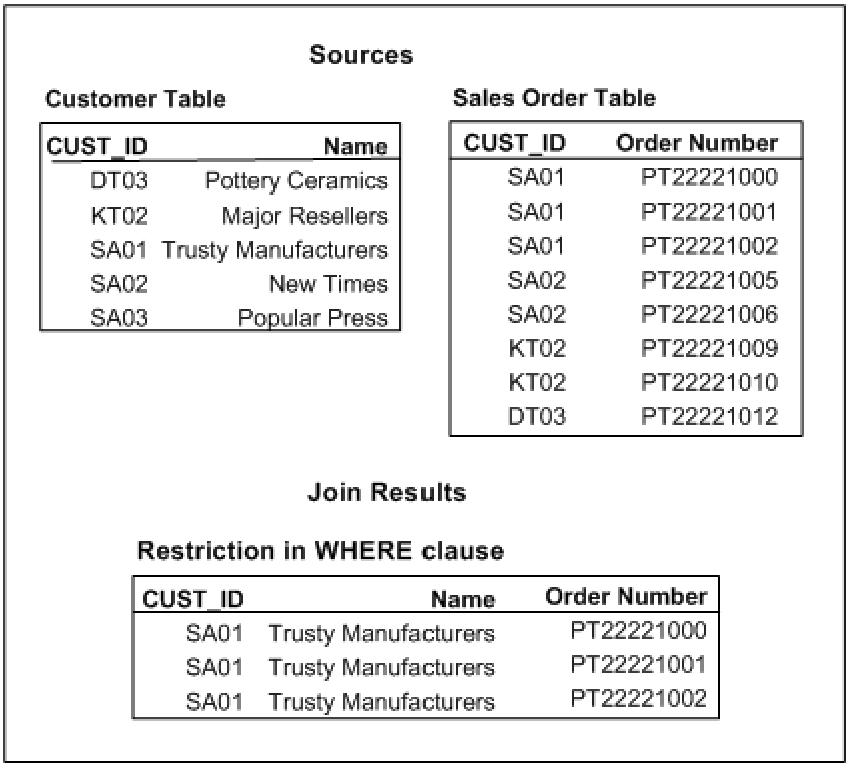
Left	Join Type	Right			Join Condition
ODS_CUSTOMER	Left outer join	ODS_SALESORDER	...		ODS_SALESORDER.CUST_ID = ODS_CUSTOMER.CUST_ID and ODS_CUSTOMER.CUST_ID = 'SA01'

The SQL query is:

```
SELECT ODS_CUSTOMER.CUST_ID, ODS_CUSTOMER.NAME1,
 ODS_SALESORDER.SALES_ORDER_NUMBER
FROM DBO.ODS_CUSTOMER ODS_CUSTOMER LEFT OUTER JOIN
 DBO.ODS_SALESORDER ODS_SALESORDER
ON (ODS_SALESORDER.CUST_ID=ODS_CUSTOMER.CUST_ID)
AND (ODS_CUSTOMER.CUST_ID = 'SA01')
```

Restriction placed in the WHERE clause example

Referring again to the example of CUSTOMER as the left source and SALESORDER as the right source of a left outer join, if the restriction `CUSTOMER.CUST_ID = 'SA01'` is placed in the WHERE clause, the result of a left outer join returns only the rows for which the restriction is true. The data in the source tables and results of the join are shown in the diagram below.



Sources

Customer Table

CUST_ID	Name
DT03	Pottery Ceramics
KT02	Major Resellers
SA01	Trusty Manufacturers
SA02	New Times
SA03	Popular Press

Sales Order Table

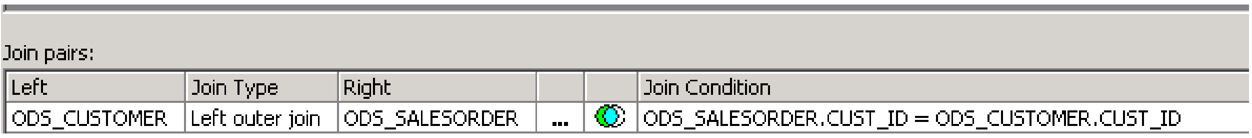
CUST_ID	Order Number
SA01	PT22221000
SA01	PT22221001
SA01	PT22221002
SA02	PT22221005
SA02	PT22221006
KT02	PT22221009
KT02	PT22221010
DT03	PT22221012

Join Results

Restriction in WHERE clause

CUST_ID	Name	Order Number
SA01	Trusty Manufacturers	PT22221000
SA01	Trusty Manufacturers	PT22221001
SA01	Trusty Manufacturers	PT22221002

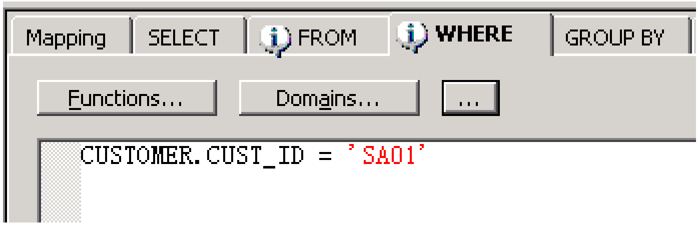
The Join pairs area of the FROM tab would appear as follows:



Join pairs:

Left	Join Type	Right			Join Condition
ODS_CUSTOMER	Left outer join	ODS_SALESORDER	...	◐	ODS_SALESORDER.CUST_ID = ODS_CUSTOMER.CUST_ID

and the WHERE tab would contain the restriction:



```
CUSTOMER.CUST_ID = 'SA01'
```

The SQL query is:

```
SELECT ODS_CUSTOMER.CUST_ID, ODS_CUSTOMER.NAME1,
 ODS_SALESORDER.SALES_ORDER_NUMBER
FROM DBO.ODS_CUSTOMER ODS_CUSTOMER LEFT OUTER JOIN
 DBO.ODS_SALESORDER ODS_SALESORDER
ON (ODS_SALESORDER.CUST_ID=ODS_CUSTOMER.CUST_ID)
WHERE (ODS_CUSTOMER.CUST_ID = 'SA01')
```

3.5.5.4.10.4 Viewing Optimized SQL

Before running a job, you can view the SQL code that SAP Data Services generates for table sources in data flows. By examining the SQL code, you can verify that the software generates the commands you expect. If necessary, you can alter your design to improve the data flow.

To view the SQL code:

1. Validate and save data flows.
2. Open a data flow in the workspace.
3. Select *Display Optimized SQL* from the *Validation* menu.

   Alternately, you can right-click a data flow in the object library and select *Display Optimized SQL*.

   The *Optimized SQL* window opens and shows a list of datastores and the optimized SQL code for the selected datastore. By default, the *Optimized SQL* window selects the first datastore.

   The software only shows the SELECT generated for table sources and INSERT INTO... SELECT for targets. It does not show the SQL generated for SQL sources that are not table sources, such as:

   - Lookup function
   - Key_generation function
   - Key_Generation transform
   - Table_Comparison transform

4. Select a name from the list of datastores on the left to view the SQL that this data flow applies against the corresponding database or application.

> **Note**
>
> The *Optimized SQL* window displays the existing SQL statement in the repository. If you changed your data flow, save it so that the *Optimized SQL* window displays your current SQL statement.

3.5.5.4.10.5 Outer join compared to the Lookup function

You can produce a similar data set using the lookup function. However, the lookup function is limited to the following actions:

Lookup function	Left Outer join
Returns only one column value for each comparison	Returns all column values for each comparison
Can be used against only one source at a time	Provides similar capability to multiple lookup calls
Cannot be used to produce the same results as non-equality joins (for example, `A.x<B.y`)	Allows non-equality joins
Permits default values other than nulls	
Can be cached when desired	

In addition, sources in an outer join query must be joined in a hierarchical order:

- A source can only be the inner table of one outer join
- A source cannot be "outer joined" with itself in a single Query transform

3.5.5.5 Row_Generation

Produces a data set with a single column. The column values start with the number that you set in the *Row number starts at* option. The value then increments by one to a specified number of rows.

3.5.5.5.1 Data inputs

None.

3.5.5.5.2 Editor

The Row_Generation transform editor includes the target schema, and transform options.

3.5.5.5.3 Options

Option	Description
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. Options are: ● *Yes*: The source is always cached unless it is the outer-most source in a join. ● *No*: The source is never cached. The default is *Yes*. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. For new jobs, specify the cache only in the Query transform editor.
Join rank	Indicates the rank of the output data set relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. For new jobs, specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.
Row count	A positive integer indicating the number of rows in the output data set. For added flexibility, you can enter a global variable or substitution parameter.
Row number starts at	An integer with which row numbering starts. If you set this option to 1, the first row will be labeled 1, the second row will be labeled 2, and so on. If you leave this blank, numbering will start at zero (0). For added flexibility, you can enter a global variable or substitution parameter.

3.5.5.5.4 Data outputs

The Row_Generation transform produces a data set with a single column and the number of rows specified in the *Row count* option. The rows contain integer values in sequence starting from the value that you entered in the *Row number starts at* option, and incrementing by one in each row.

3.5.5.6 SQL

Performs the indicated SQL query operation.

Use this transform to perform standard SQL operations when other built-in transforms cannot perform them.

The options for the SQL transform include specifying a datastore, join rank, cache, array fetch size, and entering SQL text.

> Note
>
> The SQL transform supports a single `SELECT` statement only.

3.5.5.6.1 Data inputs

None. This transform does not allow an input data set.

3.5.5.6.2 Options

Option	Description
Array fetch size	Indicates the number of rows retrieved in a single request to a source database. The default value is 1000. Higher numbers reduce requests, lowering network traffic, and possibly improve performance. The maximum value is 5000. This option is available for source tables from DB2, Informix, ODBC, Oracle, and SQL Server datastores. When retrieving a column with an Oracle long data type, Data Services automatically sets Array Fetch Size to 1. If a column has an Oracle long data type, Data Services can only retrieve one row at a time.

Option	Description
Cache	Indicates whether the software should load output from the transform into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache output when it is used in a subsequent transform as an inner source in a join. Options are: ● *Yes*: The source is cached. ● *No*: The source is not cached. The default is *Yes*. Cache specified in the Query transform editor FROM tab overrides any cache specified in a previous transform within the data flow. For new jobs, specify the cache only in the Query transform editor.
Database type	Database type and versions available in the selected datastore. The *Database type* box allows you to quickly set SQL transform values in data flows if you have multiple configurations in a datastore. This option also allows you to supply unique SQL text for each database type and version in any SQL transform instance. When you select a database type, Data Services provides the value you set previously for SQL text in that particular database type. To add or remove items in the *Database type* list box, edit the datastore configuration information using the Datastore Editor. The following describes how Data Services determines SQL text values. For more information about the Datastore Editor and its dialogs, see *Datastore editor* [page 861]. ● If the datastore has more than one configuration and there are different database types or versions, then Data Services determines the initial SQL text values for the additional database types and versions from the *Use values from* box in the Create New Configuration dialog box (a sub-dialog of the Datastore Editor). Note Join rank, Cache, and Array fetch size values remain the same as those set in the initial configuration. You cannot have more than one set of these values in a datastore. ● If you also select the *Restore values if they already exist* check box in the Create New Configuration dialog box, Data Services looks for previously defined values that once existed for that database type or version. It is possible for a data flow to contain SQL transform values for a database type or version, even if its datastore configuration was deleted. Data Services retains all SQL transform values saved with every datastore configuration. If such values exist, then Designer restores those values. Otherwise, it gets the values from the configuration you select from the *Use values from* option. If the SQL text in a SQL transform is not correct for the database type, modify the SQL text. If the SQL text has any hard-coded owner names or database names in it, consider replacing them with variables to limit the number of modifications you need for new database types.

Option	Description
	Note Because Data Services only provides values for variables during run time, do not use variables in the SQL text of a SQL transform when you use the *Update Schema* button. To support portability, add variables afterwards.
Datastore	The name of the datastore that Data Services uses to access the tables referred to in *SQL text*.
Join rank	Indicates the rank of the output data set relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. For new jobs, specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.
SQL text	The text of the SQL query. This string is passed to the database server. You do not need to put enclosing quotes around the SQL text. You can put enclosing quotes around the table and column names, if required by the syntax rules of the DBMS involved.
Update schema	Click this option to automatically calculate and populate the output schema for the SQL SELECT statement.

3.5.5.6.3 Data outputs

There are two ways of defining the output schema for a SQL transform:

- Automatic: After you type the SQL statement, click *Update schema* to execute a described select statement against the database which obtains column information returned by the select statement and populates the output schema.
- Manual: Output columns must be defined in the output portion of the SQL transform.
  The number of columns defined in the output of the SQL transform must equal the number of columns returned by the SQL query.
  The column names and data types of the output columns need not match the column names or data types in the SQL query. Data Services conversion rules apply.
  When possible, Data Services optimizes data flows by pushing expressions down to an underlying database manager. In a single transaction, Data Services can push down expressions so that they are performed by the underlying database manager. However, when Data Services evaluates an expression which includes operands of more than one data type, Data Services attempts to convert the operands to the same data type first. (Except for national character-set data types which can be pushed down while others in an expression are not.). Errors are flagged for illegal conversion.
  The output data set cannot contain hierarchical data.

Exercise care when specifying the output columns. Typically the column data types of the two sets of columns should be an exact match. If you choose to have different data types, you need to ensure that they are compatible—if they are not, you will get a runtime error from the underlying database manager.

Related Information

varchar [page 1038]

3.5.5.7 Validation

The Validation transform qualifies a data set based on rules for input schema columns. You can apply multiple rules per column or bind a single reuseable rule (in the form of a validation function) to multiple columns.

The Validation transform can identify the row, column, or columns for each validation failure. You can also use the Validation transform to filter or replace (substitute) data that fails your criteria.

When you enable a validation rule for a column, a check mark appears next to it in the input schema.

3.5.5.7.1 Validation Rules tab options

Open the Validation transform editor by clicking the name of the transform in your data flow. On the Validation Rules tab, the top pane lists all of available rules and the bottom pane lets you define substitution values for failed rules to send to the Pass output schema.

The following table describes the options for the Validation Rules tab.

Type	Option	Description
Rules buttons	Add	Click to add a new rule. Launches the *Rule Editor* dialog box.
	Edit	Select an existing rule and click to edit the rule. You can also double-click a rule to open the Rule Editor.
	Remove	Select one or more rows and click to delete the rule(s).
Rules columns	Enabled	Select to enable the rule; clear to disable it. ou can also enable or disable a rule in the Rule Editor dialog box.

Type	Option	Description
	Rule	The syntactical name of the rule.
	Ignore if NULL	If set to Yes, Data Services skips (ignores) the rule if any of the associated column values are NULL. You set this option in the Rule Editor.
	Action on Fail	Identifies what action to take when the row fails: send the row to the Fail target, Pass target, or both. If you choose *Send to Pass* or *Send to Both*, you can choose to substitute a value or expression for the failed values that go to the Pass output using the bottom pane of the Validation Rules tab.
	Name	The common name as defined in the *Name* field of the Rule Editor.
	Description	An optional description as defined in the Rule Editor.
If rule(s) fails and send to Pass, substitute:	Remove button	Select one or more rows and click to delete the substitution(s).
	Enabled	Select to enable the substitution; clear to disable it.
	Column	Identifies the column to which the substitution will apply. Double-click the cell to enable the drop-down list.
	Expression	Defines the substitution constant, variable, or function call. Double-click the cell to type in the cell. During job execution, Data Services converts substitute values to a corresponding column data type: integer, decimal, varchar, date, datetime, timestamp, or time.
	Ellipses button	As an alternative to typing a value in the Expression cell, click the ellipses button to launch the Smart Editor, where you can configure variables, substitution parameters, and functions for defining the substitution.

Related Information

Validation Transform Options tab [page 1474]

3.5.5.7.2 Validation Transform Options tab

The Validation Transform Options tab has the following options:

Option		Description
On failure:	Collect data validation statistics	Select this option to generate statistics for columns that failed validation to view in the Data Validation dashboards metadata reports.
	Collect sample data	Select this option to capture sample data for columns that failed validation to view in Data Validation dashboard metadata reports.
Output Rule Violation Information	Create column DI_ROWID on Validation_Fail	Select to include a DI_ROWID column in the Fail output schema (selected by default).

Related Information

Validation Rules tab options [page 1472]

3.5.5.7.3 Rule Editor

You use the Rule Editor to define or edit a validation rule. Launch the Rule Editor from the Validation Rules tab of the Validation transform by clicking *Add* or double-clicking an existing rule to edit it.

The Rule Editor lets you configure a rule either based on an existing validation function or a rule defined in the validation transform that is based on a single column input.

Rules defined with a validation function have the following characteristics:

- Reusable
- Accept multiple input parameters
- Prefereble for more complex rules
- Can be created in SAP Information Steward and imported and used in Data Services

Rules defined with a column validation have the following characteristics:

- Not reusable; the rule definition is part of the transform (not shared outside the transform)
- Binds to only one input column
- Better for simple rules

Also note there are two types of validation functions as categorized in the object library:

- Imported from Information Steward: These functions were created in Information Steward and cannot be edited in Data Services
- Locally created: Created and editable in Data Services

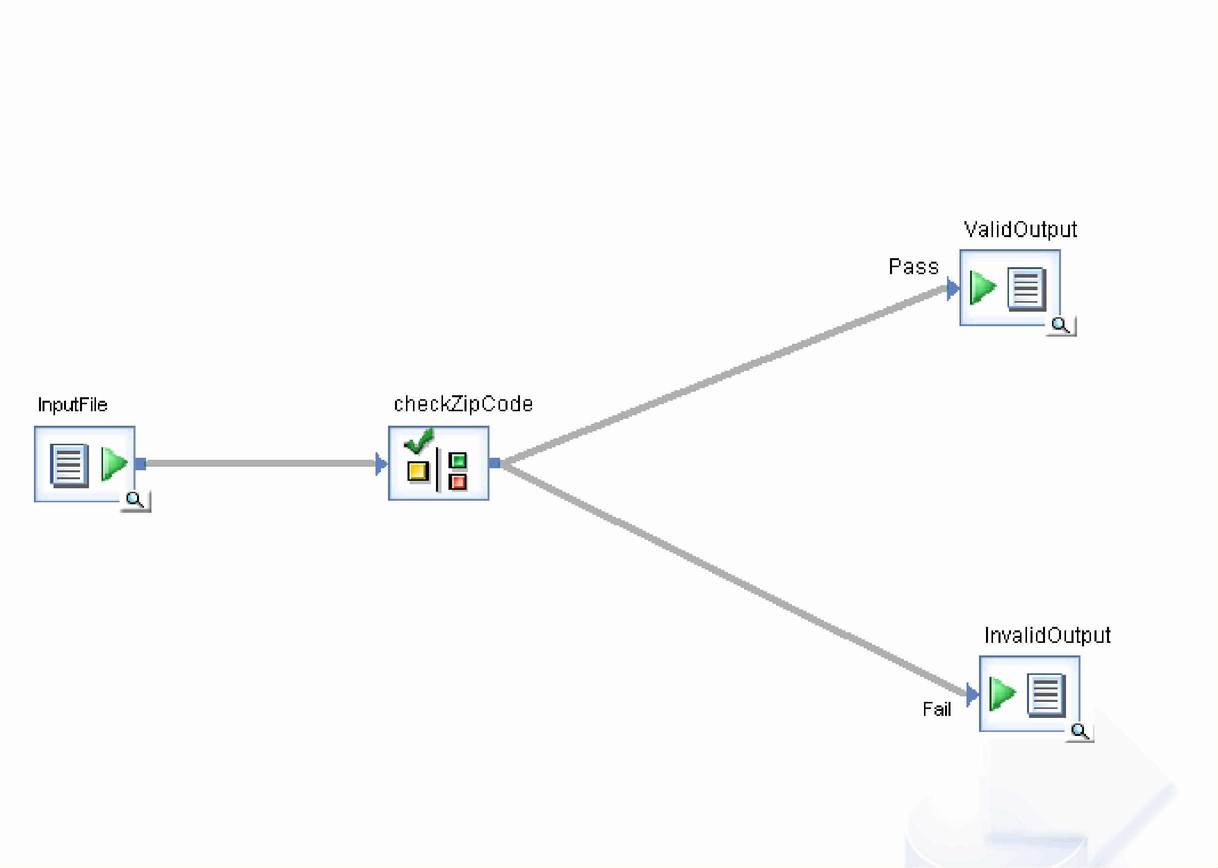
3.5.5.7.3.1 To define a validation rule based on a column

1. In a data flow with the Validation transform connected to an input schema, click the transform name to open its editor.
2. On the Validation Rules tab, click *Add*.
   The Rule Editor displays with the *Enabled* check box selected by default.
3. Type a name for the rule and optionally add a description.
4. Select an *Action on fail*:
   - Send To Fail
   - Send To Pass
   - Send To Both
5. Select *Column Validation*.
6. Select a *Column* from the drop-down list.
7. Define a condition. All conditions must be Boolean expressions. Or, select *Custom Condition* to enable access to the smart editor (ellispses button) or function wizard (*Functions* button).

Example

Defining a validation rule for a five-digit ZIP code pattern

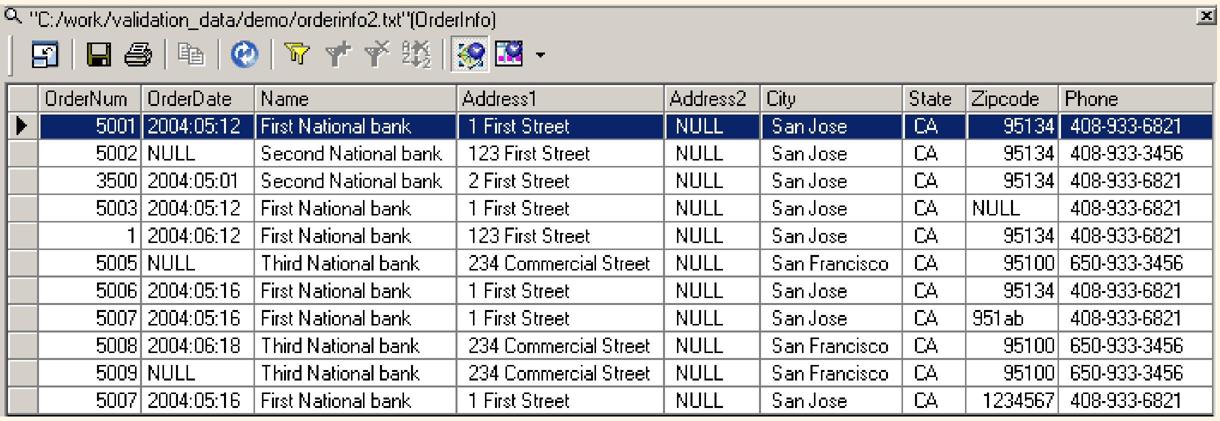
The following example defines a Validation transform that verifies that the data for ZIP code is a five-digit number.



1. Open the Validation transform editor.
2. Click *Add*.
3. Name the rule **checkzip**.
4. For *Action on Fail*, select *Send To Both*.
5. Select *Column Validation*.
6. Select the *Customers.Zipcode* column.
7. Select the *LIKE* condition.
8. Specify the pattern as **99999**, which matches any five-digit number.
9. Click *OK*.
10. On the *Validation Rules* tab, for the option, *If any rule fails and send to Pass, substitute with:* double-click a cell under *Column* and select *Customers.Zipcode*.
11. Double-click in the cell under *Expression* and type **'Invalid'**.

If the zip code does not match the pattern, Data Services replaces the value with the string **Invalid**.

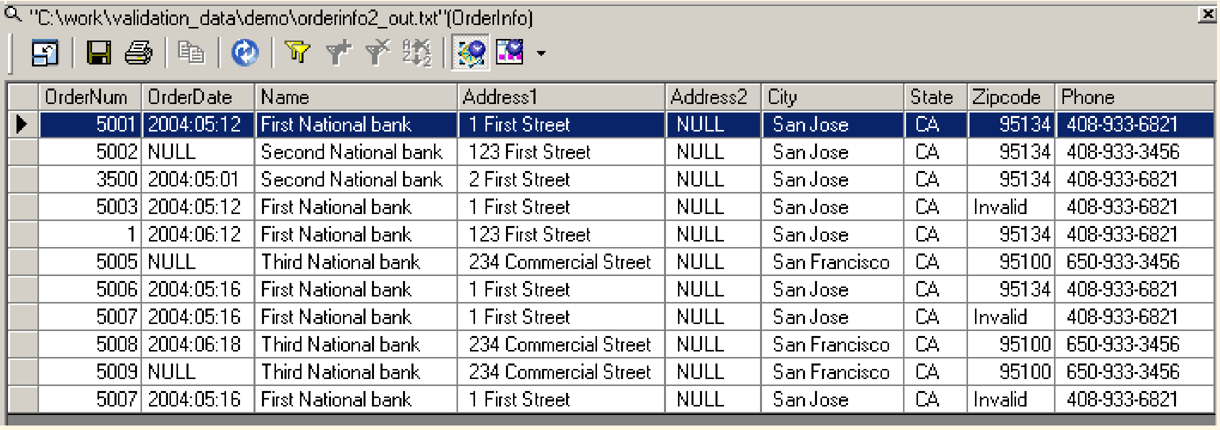
The following graphic shows the input data.



"C:/work/validation_data/demo/orderinfo2.txt"(OrderInfo)

OrderNum	OrderDate	Name	Address1	Address2	City	State	Zipcode	Phone
5001	2004:05:12	First National bank	1 First Street	NULL	San Jose	CA	95134	408-933-6821
5002	NULL	Second National bank	123 First Street	NULL	San Jose	CA	95134	408-933-3456
3500	2004:05:01	Second National bank	2 First Street	NULL	San Jose	CA	95134	408-933-6821
5003	2004:05:12	First National bank	1 First Street	NULL	San Jose	CA	NULL	408-933-6821
1	2004:06:12	First National bank	123 First Street	NULL	San Jose	CA	95134	408-933-6821
5005	NULL	Third National bank	234 Commercial Street	NULL	San Francisco	CA	95100	650-933-3456
5006	2004:05:16	First National bank	1 First Street	NULL	San Jose	CA	95134	408-933-6821
5007	2004:05:16	First National bank	1 First Street	NULL	San Jose	CA	951ab	408-933-6821
5008	2004:06:18	Third National bank	234 Commercial Street	NULL	San Francisco	CA	95100	650-933-3456
5009	NULL	Third National bank	234 Commercial Street	NULL	San Francisco	CA	95100	650-933-3456
5007	2004:05:16	First National bank	1 First Street	NULL	San Jose	CA	1234567	408-933-6821

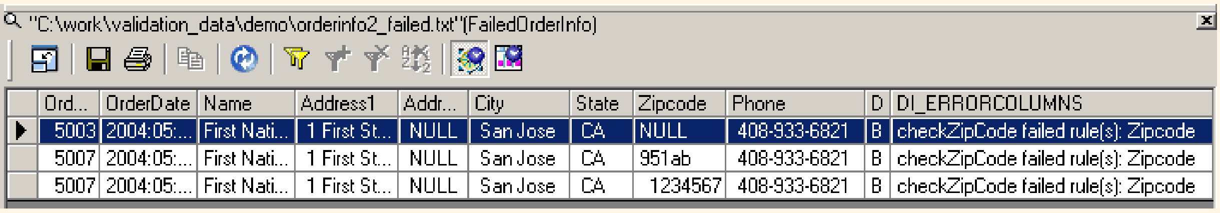
The following graphic shows the data sent to the Pass/ValidOutput table:



"C:\work\validation_data\demo\orderinfo2_out.txt"(OrderInfo)

OrderNum	OrderDate	Name	Address1	Address2	City	State	Zipcode	Phone
5001	2004:05:12	First National bank	1 First Street	NULL	San Jose	CA	95134	408-933-6821
5002	NULL	Second National bank	123 First Street	NULL	San Jose	CA	95134	408-933-3456
3500	2004:05:01	Second National bank	2 First Street	NULL	San Jose	CA	95134	408-933-6821
5003	2004:05:12	First National bank	1 First Street	NULL	San Jose	CA	Invalid	408-933-6821
1	2004:06:12	First National bank	123 First Street	NULL	San Jose	CA	95134	408-933-6821
5005	NULL	Third National bank	234 Commercial Street	NULL	San Francisco	CA	95100	650-933-3456
5006	2004:05:16	First National bank	1 First Street	NULL	San Jose	CA	95134	408-933-6821
5007	2004:05:16	First National bank	1 First Street	NULL	San Jose	CA	Invalid	408-933-6821
5008	2004:06:18	Third National bank	234 Commercial Street	NULL	San Francisco	CA	95100	650-933-3456
5009	NULL	Third National bank	234 Commercial Street	NULL	San Francisco	CA	95100	650-933-3456
5007	2004:05:16	First National bank	1 First Street	NULL	San Jose	CA	Invalid	408-933-6821

The following graphic shows the data sent to the Fail/InvalidOutput table:



"C:\work\validation_data\demo\orderinfo2_failed.txt"(FailedOrderInfo)

	Ord...	OrderDate	Name	Address1	Addr...	City	State	Zipcode	Phone	D	DI_ERRORCOLUMNS
▶	5003	2004:05:...	First Nati...	1 First St...	NULL	San Jose	CA	NULL	408-933-6821	B	checkZipCode failed rule(s): Zipcode
	5007	2004:05:...	First Nati...	1 First St...	NULL	San Jose	CA	951ab	408-933-6821	B	checkZipCode failed rule(s): Zipcode
	5007	2004:05:...	First Nati...	1 First St...	NULL	San Jose	CA	1234567	408-933-6821	B	checkZipCode failed rule(s): Zipcode

3.5.5.7.3.2 Defining a validation rule using a custom validation function

This section describes how to first add and define the new custom validation function in the object library, then how to use the function to define a validation rule.

3.5.5.7.3.2.1 To define a custom validation function in the object library

This procedure describes how to add and define the a custom validation function in the object library. The next procedure describes how to use the validation function to define a validation rule.

In this example, the function determines whether the ZIP column contains 5-digit ZIP codes by checking each character and ensuring that each character is a digit. To verify the character is a digit, the function checks to see if its ASCII value is between 48 and 57 inclusive (corresponding to 0 through 9 respectively).

1. From the *Custom Function* tab of the object library, right-click *Validation Functions* and select *New*.
   The *Custom Function* dialog box displays with *Validation function* selected.
2. Enter the name **`ZipIsValid`** for the new function and click *Next*.
   The smart editor displays.
3. To define the parameters and variables for this function, select the *Variables* tab in the smart editor window.
4. Right-click the *Parameters* node and select *Insert*.
5. Enter a parameter name such as **`$ZipToValidate`**.
6. Select the appropriate data type such as *varchar* with a length of **`5`**.
7. For the *Parameter type* select *Input*.

   > i Note
   >
   > Validation functions can have multiple input parameters; additional output parameters are not used.

8. Click *OK*.
   The parameter $ZipToValidate appears under the *Parameters* node.
9. Add five local variables named $Z1 through $Z5, which represent the five characters in the column. Right-click *Local*, select *Insert*, enter the name, select the data type *int*, and click *OK*.

The five variables appear under the *Local* variables node.

10. In the text panel of the smart editor, enter the following validation script:

```
$Z1 = ascii(substr($ZipToValidate, 1, 1));
$Z2 = ascii(substr($ZipToValidate, 2, 1));
$Z3 = ascii(substr($ZipToValidate, 3, 1));
$Z4 = ascii(substr($ZipToValidate, 4, 1));
$Z5 = ascii(substr($ZipToValidate, 5, 1));

if ( $Z1 < 48 OR $Z1 > 57 )
 return 0;
if ( $Z2 < 48 OR $Z2 > 57 )
 return 0;
if ( $Z3 < 48 OR $Z3 > 57 )
 return 0;
if ( $Z4 < 48 OR $Z4 > 57 )
 return 0;
if ( $Z5 < 48 OR $Z5 > 57 )
 return 0;

return 1;
```

> Note
>
> The Validation transform only supports custom functions that return an integer data type. If a return value is not a zero, then Data Services processes it as TRUE.

11. Click *OK*.
    The new function displays in the object library under the *Validation Function* node.

Continue to the next procedure to use the validation function in a validation rule.

3.5.5.7.3.2.2 To define a validation rule using a validation function

This procedure describes how to use a custom validation function to define a validation rule.

1. Add a data flow with a source and a Validation transform and connect them.
2. Click the transform name to open the Validation transform editor.
3. On the *Validation Rules* tab, click *Add*.
   The Rule Editor displays with the *Enabled* check box selected by default.
4. In the Rule Editor, name the rule.
5. Select the *Action on fail: Send to Fail*.
6. With the *Validation Function* option selected, from the drop-down list select the ZipIsValid function.
   The Bindings table populates with the required parameters for the function.
7. Define the parameter argument. For the $ZipToValidate parameter, double-click the cell under *Argument* and select the column *ZIP*.
8. Click *OK* to save the rule.
   The rule appears in the Rules list on the *Validation Rules* tab.

When you run the job, the rule will apply the reusable validation function to the ZIP column and check each character to validate each is a digit.

3.5.5.7.3.3 Rule Editor options

The following table describes the options in the Rule Editor dialog box.

Option	Description
Name	The rule name as created by the user.
Description	An optional descripion provided by the user.
Enabled	Select to apply the rule when running the job. Clear to disable the rule. You can also enable or disable the rule on the *Validation Rules* tab.
Ignore if NULL	Select to not apply the validation rule for any incoming values that are NULL, and send all NULL values to the Pass data output.
Action on Fail	Identifies what action to take when the row fails: send the row to the Fail target, Pass target, or both. If you choose *Send to Pass* or *Send to Both*, you can choose to substitute a value or expression for the failed values that go to the Pass output in the bottom pane of the *Validation Rules* tab.
Validation Function	Select to define the rule based on a validation function. The function must have already been created in the object library before it will be available in the drop-down list.
Bindings	Each parameter required by the function displays. Provide an argument for each parameter. The argument can be a constant, column (from the input schema), substitution variable, data flow variable, or global variable. You can type the value into the cell, or select the drop-down arrow to display the available columns and variables. Clear the *Score* check box to not include the binding in the Fail output. See the following example for more information about Score.
Column Validation	Select to base the rule on a single input column.
Column	Click the drop-down arrow to select a column on which to process the rule.
Condition	Select a condition (and usually in conjunction with an expression) to define the column-based rule. Available condition operators include: *<, >, <=, >=, <>, =* *IS NULL, IS NOT NULL* *LIKE* *IN SET* *BETWEEN*: Use to specify a range of values.

Option	Description
	Match Pattern: Lets you enter a pattern based on the Data Services match_pattern function. *Exists in Table*:Select to specify that a column's value must exist in another table's column. Click the drop-down arrow to open the *Input Parameter* window and browse to the desired input. This option uses the LOOKUP_EXT function. Define the NOT NULL constraint for the column in the LOOKUP table to ensure the *Exists in table* condition executes properly. *Custom Condition*: Select to create more complex expressions by linking to the smart editor (ellipses button) or function wizard (Function button). An edit box opens for you to enter your expression.

Example

Understanding Score

Say you are using a validation function to define a validation rule that checks for valid ZIP codes and has two input parameters: $Country and $ZIP. The function might be written as follows:

```
 if ($Country = 'US')
  if (match_pattern ($ZIP,'99999')=1)
   return 1; #For the country US, the ZIP code must match the 5-digit pattern
  else
  return 0; #Fails if ZIP code does not match pattern for US
return 1; #If country is not US, always true
```

Using this function in a Validation transform binds the rule to the columns Country and ZIP. However in the Rule Editor when defining the parameter, you would select the *Score* check box for the ZIP column but not for the Country column because the goal is to validate the ZIP format. The $Country parameter is being used as a filter (to sort out U.S. ZIP codes), not for validation.

Sample input:

Row_ID	Country	ZIP
1	US	12345
2	US	123
3	Belgium	123

Fail output with Score selected for ZIP:

Row_ID	Country	ZIP	DI_ERROR_ACTION	DI_ERROR_COLUMNS
2	US	123	F	Validation failed rule(s): ZIP

RuleViolation output:

Row_ID	DI_RULENAME	DI_COLUMNNAME
2	IsValidZip	ZIP

Therefore, if you later want to generate a report on why rows failed, only the ZIP column will appear as having has bad data, not the Country column.

Related Information

match_pattern [page 1622]
lookup_ext [page 1605]

3.5.5.7.4 Data outputs

The Validation transform can output up to three data outputs: Pass, Fail, and RuleViolation. Data outputs are based on the condition that you specify in the transform. You set the data outputs when you connect the output of the Validation transform with a Pass object, a Fail object, or both a Pass and Fail object in the workspace.

You can also load Pass and Fail data into multiple targets.

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Pass</td><td>When you choose a Pass type data output, the output schema is identical to the input schema.</td></tr>
<tr><td>Fail</td><td>When you choose a Fail type data output, Data Services adds the following columns to the Fail output schema: DI_ERRORACTION and DI_ERRORCOLUMNS.
<ul>
<li>DI_ERRORACTION: This column indicates where Failed data was sent:
<table>
<tr><th>Indicator</th><th>Description</th></tr>
<tr><td>B</td><td>Sent to both Pass and Fail outputs.</td></tr>
<tr><td>F</td><td>Sent to the Fail output.</td></tr>
</table>
Note
If you choose to send failed data to the Pass data output, Data Services does not track the results. Because Data Services does not add columns to Pass output types, you may substitute a value for failed data that you send to the Pass data output. The input schema is maintained in the Pass output.</li>
<li>DI_ERRORCOLUMNS: This column displays all error messages for columns with failed rules. The names of input columns associated with each message are separated by colons. For example: <ValidationTransformName> failed rule(s): c1:c2</li>
</ul></td></tr>
</table>

Option	Description
	ℹ Note If a row has conditions set for multiple columns and the Pass, Fail, and Both actions are specified for the row, then the precedence order is Fail, Both, Pass. For example, if one column's action is Send to Fail and the column fails, then the whole row is sent only to the Fail output. Other actions for other validation columns in the row are ignored.
RuleViolation	You use the RuleViolation output to capture each row that represents the unsuccessful execution of a validation rule. The following output schema columns contain data to help you understand which rule failed: ● DI_ROWID: Multiple validation rules can fail on a single input row; however, the output schema Fail only emits a single row to report each failure. To get complete information about every failed rule, this column associates rows sent to the Fail output to those recorded in the RuleViolation output. To link the DI_ROWID to the actual data row, on the Validation Transform Options tab make sure the *Create column DI_ROWID on Validation_Fail* is selected, which includes DI_ROWID in the Fail output schema. ● DI_RULENAME and DI_COLUMNNAME: You can associate columns with multiple rules, and a rule can be associated with multiple columns, so each row identifies both the rule name and the column name so you can uniquely identify a particular rule-column pair. Only columns with *Score* selected in the Rule Editor will be added.

3.5.5.7.5 Nested schemas

The Validation transform can be used with nested schemas. You can associate any scalar column in a nested schema with a validation rule. You can use other nested columns in a validation condition as long as they share the same parents with the column on which the rule is defined. Data Services generates additional columns (DI_ERRRORACTION and DI_ERRORCOLUMNS) for the Fail output target at the top level of the schema only. Columns must be expressed with fully qualified names.

3.5.5.8 XML_Map

The XML_Map transform is a data transform engine designed for hierarchical data. It provides functionality similar to a typical XQuery or XSLT engine. The XML_Map transform takes one or more source data sets and produces a single target data set. Flat data structures such as database tables or flat files are also supported as both source and target data sets. You can use the XML_Map transform to perform a variety of tasks. For example:

- You can create a hierarchical target data structure such as XML or IDoc from a hierarchical source data structure.
- You can create a hierarchical target data structure based on data from flat tables.
- You can create a flat target data set such as a database table from data in a hierarchical source data structure.

XML_Map transform works in two modes- Normal and Batch mode. In normal mode, data is handled on a row by row basis before sending it to the next transform. In batch mode, data is handled as block of rows, before sending it to the next transform. There are different transform icons to indicate each mode.

3.5.5.8.1 Data inputs

Normal mode - One or more data sets. Each data set can be a hierarchical data source such as XML, IDoc, or a hierarchical output structure from a previous transform. It can also be row-based data such as a database table, spreadsheet, or flat file.

Batch mode - Single data set accepted. The data set can only be a hierarchical data source such as XML, IDoc, or a hierarchical output structure from a previous transform. It can also be row-based data such as a database table, spreadsheet, or flat file.

3.5.5.8.2 Data outputs

Normal mode - A single data set. The data set may be a hierarchical structure or row-based flat data.

Batch mode - A single data set. The data set is always a hierarchical structure or flat dataset. The Batch key columns become first-level columns of the row. The rest of the columns become second-level columns and cannot be edited.

3.5.5.8.3 Editor

Use the XML_Map editor to specify the Schema In, Schema Out, and Options for the XML_Map transform.

The areas can be resized in order to expand the area in which you are working. You can also expand and contract the columns to change the width of properties displayed in the input and output schema areas.

3.5.5.8.4 Options

Use batch mode to accumulate blocks of rows before you send those rows to the next transform. For example, using batch mode with SAP functions may improve performance.

You can group the data by *Batch size* and optionally by the *Batch key columns* in batch mode.

Options	Description
Batch size	This option specifies the maximum size of second-level rows for each batch. It could be a positive number, a global variable, or a substitution parameter. The default value of batch size is 10000.

Options	Description
Batch key columns	This option specifies the input columns on which a given batch is constructed. Batch key columns become root-level columns of the ouptut schema. When batch key column is selected, "*Input already sorted by batch key columns*" checkbox option is available. Checking it indicates that input is sorted, so the transform doesn't need to do sorting itself. Note If the Batch key column is not specified, then *Batch size* defines the size of the block of rows.

3.5.5.8.5 Batch mode operation

When the XML_Map transform is run in batch mode, the number of rows in the first nested level is grouped by the *Batch size* and optionally by the *Batch key columns*.

When the *Batch key columns* option is selected, the input for the XML_Map transform is sorted based on *Batch key columns*. Sorting is done to detect batch key changes. If the *Batch key columns* option is not selected, then there is no sort operation. Each batch is constructed purely based on the maximum *Batch size*. There are no first-level columns.

Based on the type of input, each batch is constructed. The batch is sent out when it reaches the maximum *Batch size*, and optionally if the *Batch key* changes.

3.5.5.8.5.1 Using Batch mode in XML_Map transform

Follow these steps to configure the XML_Map transform for batch mode.

> Note
>
> Only one input is allowed for the XML_Map transform in batch mode.

1. In the XML_Map transform editor, select *Batch Mode*.

   > Note
   >
   > A special symbol indicates that the XML_Map transform is in batch mode.

2. Construct each batch based on the *Batch size*, and optionally the *Batch key columns*.
3. Select the maximum *Batch size* value of second level rows for each batch.

   > Note
   >
   > Default value is 10000 and it could be a positive number, a global variable, or a substitution parameter.

4. If you opt for *Batch key columns*, then drag the selected column name from the input schema into the *Batch key columns* list box.

   > ℹ Note
   >
   > - The whole input schema moves one level down in the output schema, making it nested.
   > - Batch key columns become part of the output schema at the top-level. However, if you want to modify the output schema, then add downstream transforms.
   > - If the Batch key columns are not selected, then there is no sort operation. Each batch is constructed purely based on maximum batch size. There are no first-level columns.

5. Select the checkbox *input already sorted by batch key columns* if you need to indicate that input is sorted.

The batch is sent out when it reaches the maximum batch size, and optionally if the batch key changes.

3.5.5.8.6 Searching in an input or output schema

1. In the XML_Map editor *Find* tab, enter the search term in the *Find what* box or select from previous search terms in the drop-down list.
2. In the *Schemas* list, choose the schemas in which to search.
3. In the *Elements* list, choose the types of mappings in which to search.
4. In the *Where* list, choose the properties to search within.

   > ℹ Note
   >
   > You can search within one or all properties, but not within two or three specific properties at a time.

5. Select the *Match case* check box to constrain your search to the capitalization entered.
6. Click *Find*.
   The Designer searches the transform configuration for the words you specified within the constraints you defined.

   > ℹ Note
   >
   > The Designer searches for columns loaded into memory. If columns are not loaded into memory, you must expand the schema to load the columns into memory before clicking *Find* and searching for the columns.

All matches are shown in the box below the find constraints. When you click to select a table or column name, the table or column is automatically highlighted and shown in the corresponding input or output schema area.

Initially, the Designer lists the matching columns in the order that they appear within the schemas. If you are searching both schemas, the Designer lists the first match found in the input schema first and the last match found in the output schema last. You can sort the list of matches by property. Each time you click a property heading, the Designer resorts the matches, cycling through original order, ascending order, and descending order.

Arrow icons confirm column and sort type. For example, if you sort the data by the *Description* property and in ascending order, an “up” arrow appears next to the *Description* heading. Click the heading again and a “down” arrow appears to indicate that the data is now sorted in descending alpha-numeric order. Click again and the match list returns to its original sort order.

3.5.5.8.7 XML_Map transform input schema

The input schema area displays all input schemas for the XML Map transform. Each input schema can contain zero or more of the following elements:

- Columns
- Nested schemas

Icons preceding columns are combinations of the following graphics:

Icon	Description
🔑	Primary key.
▯	Column that is not used in output mapping.
▶	Column that is used in output mappings.

The *Schema In* list at the top left of the query editor indicates the schema that is currently selected. As you select schemas or columns in the input schema area, the *Schema In* list displays the corresponding schema. Conversely, you can select a schema in the *Schema In* list to move easily to a required schema.

You can right-click elements in the input schema area and select the following menu commands:

Command	Applicable elements	Effect
Copy	Columns, schemas	Stores a copy of the selected elements in the clipboard, leaving the elements in the input schema area.
Find	Anywhere in the input schema area	Locates an output element with the name or description you enter.
Refresh	Anywhere in the input schema area	Refreshes the display of the input schema area.
Parent	Columns	Selects the parent schema of the selected column.
Collapse	Columns, schemas	Collapses a selected schema or a selected column's parent schema (to facilitate viewing/navigation).
Generate DTD	Root schema only	Generates a DTD format that corresponds to the structure of the selected schema (either NRDM or relational). Generates all data types as varchar.
Generate XML Schema	Root schema only	Generates an XML Schema that corresponds to the structure of the selected schema (either NRDM or relational). All data types match those of the selected schema.
Create File Format	Schemas	Creates a file format from a relational table schema. All data types match those of the original table schema.
Create HDFS File Format	Schemas	Creates an HDFS file format from a relational table schema. All data types match those of the original table schema.
Properties	Columns, schemas	Displays the properties of the selected element. ℹ Note You cannot modify the element properties.

3.5.5.8.8 XML_Map transform output schema

The output schema area displays the schema output from the XML_Map transform. The output schema can contain one or more of the following elements:

- Columns
- Nested schemas

Icons preceding columns are combinations of the following graphics:

Icon	Description
🔑	Primary key.
▶	Column that has a simple mapping. A simple mapping is either a single column or an expression with no input column (that is, an expression that does not vary with input).
⏩	Column that has a complex mapping. A complex mapping is any mapping that is not simple.
❌	(Red cross superimposed on any icon) Incorrect mapping. ℹ Note Data Services does not perform a complete validation during design, so the editor may not flag an incorrect mapping. For a complete validation, select ▶ *Validation* > *Validate* ◀.

The Schema Out pane shows the following:

- The current schema in the Schema Out list at the top and in the output schema area. The current schema determines:
  - The output elements that you can modify (add, map, or delete).
  - The scope of the Iteration Rule through ORDER BY tabs in the options area.
- Non-current schemas appear dim in the output schema area.

3.5.5.8.9 Change the current schema

There are several ways to change the current schema in a XML_Map transform:

- Select a schema from the Output list.
- Right-click a schema or column in the output schema area and select *Make Current*.
- Double-click one of the non-current (dim) elements in the output schema area.

When you connect a target table to an XML_Map transform with an empty output schema, Data Services automatically fills the transform's output schema with the columns from the target, without mappings.

The software only fills the target schema in the output of a transform when you connect a target table to a transform with an empty output schema. If the output schema contains any column mappings, the software does not overwrite those mappings. Similarly, if you connect a transform to one target, and then disconnect that target and connect to another target, the output schema will show the columns from the first target connected.

There are several techniques to change the output schema:

- Drag and drop (or copy and paste) columns or nested schemas from the input schema area to the output schema area (this provides simple column mappings).
  If you drop a column on an existing column, you can remap that column. Select *Remap Column* to update only the column mapping or select *Remap with Data Type* to update the column mapping and data type. Alternatively, you can select *Insert Above* or *Insert Below* to add the column as a new mapping or *Cancel* if you do not want to add the column to the output schema.
- Right-click the current schema and select *New Output Column* or *New Output Schema*. You can provide simple column mappings by dragging input columns over the new output columns. For complex mappings, use the options area.
- Right-click columns in the current schema to assign and reverse primary key settings on output columns. A key icon indicates primary keys.
- Right-click the current schema and select *Unnest* to flatten output schemas. Use this command when a job has a source with a nested schema (such as an XML file), and you map columns from this source to a flat target table schema.

You can right-click elements in the output schema area and select commands. Generally, the elements must be within the current schema.

Command	Applicable elements	Effect
Cut	All	Removes the selected elements from the output schema area and stores a copy of the elements in the clipboard.
Copy	All	Stores a copy of the selected elements in the clipboard, leaving the elements in the output schema area.
Paste	All	Inserts the elements stored in the clipboard at the current cursor location (this must be within the current schema). Only visible when the clipboard contains something. If the cursor overlaps an existing column, you are prompted to insert above, insert below, remap column, or cancel. Note Copying an input element and pasting it in the output schema area provides a simple mapping from the input element to the output element. You can also do this by dragging the input element to the output schema area.
Delete	All	Removes the selected elements from the output schema area (without making a copy).
Find...	All	Locates an output element with the name or description you enter.

Command	Applicable elements	Effect
Make Current	All outside the current schema	Makes the selected schema, or the schema of the selected element, the current schema.
New Output Column...	Schemas	Adds an output column to the current schema with the name and properties you enter.
New Output Schema...	Schemas	Adds a nested schema to the current schema with the name you enter.
Propagate Column From	Columns	Carries a selected column schema from an upstream source or transform through intermediate objects to the output schema. Simple mappings are created in each object with no change to the data type or data itself.
Unnest	Nested schemas	Flattens the selected schema. The selected schema disappears and a flat list of columns appears.
Primary Key	Columns	Toggles the primary key attribute of the column on (check mark appears next to the command) or off (no check mark appears next to the command). A key icon indicates that a column is a primary key.
Optional	All	Toggles to make a schema or column optional.
Repeatable	Nested schemas	Toggles to make a schema repeatable.
Generate DTD	Root schema only	Generates a DTD format that corresponds to the structure of the selected schema (either NRDM or relational). Generates all data types as varchar.
Generate XML Schema	Root schema only	Generates an XML Schema that corresponds to the structure of the selected schema (either NRDM or relational). All data types match those of the selected schema.
Create File Format	Nested schemas	Creates a file format from a relational table schema. All data types match those of the original table schema.
Create HDFS File Format	Nested schemas	Creates an HDFS file format from a relational table schema. All data types match those of the original table schema.
Properties	All	Displays the properties of the selected element.

3.5.5.8.10 Configure mappings

The mapping configuration area of the XML_Map editor contains several tabs where you enter information to specify the data you want retrieved. Tabs containing entries are flagged by a special ℹ icon.

When you drag and drop (or copy and paste) input columns to the output schema, Data Services inserts a value in the Mapping tab. For simple mappings, this may be sufficient. For more complex mappings, complete the appropriate tabs.

Tab	Description
Mapping	Specifies how the selected output column will be derived (or mapped).
Iteration Rule	Specifies how instances for the current schema are created from instances of the source(s). An iteration rule can only be created for a repeatable target schema. Additionally, in most situations, a repeatable target schema must have an iteration rule.

WHERE	Specifies conditions that determine which instances of the target schema are output.
	Enter the conditions in SQL syntax, like a WHERE clause in a SQL SELECT statement. For example:
	`TABLE1.EMPNO = TABLE2.EMPNO`
	`AND TABLE1.EMPNO > 1000`
	`OR TABLE2.EMPNO < 9000`
	Use the *Functions*, *Domains*, and *smart editor* buttons for help building expressions.

DISTINCT	Specifies the list of distinct columns from the input or output schema (if required). **Restriction** You can not mix input and output columns in the DISTINCT list.
GROUP BY	Specifies how the output instances are combined (if required).
ORDER BY	Specifies how the output instances are sorted (if required).
Advanced	Specifies whether to run the transform in a separate process, and defines additional options for input schemas. Options set in the Advanced tab apply to the entire XML_Map transform.
Find	Enables you to search for a specific word or term in the input and/or output schemas.

Note

Use the WHERE through ORDER BY tabs to specify additional constraints for the current schema, similar to SQL SELECT statement clauses.

3.5.5.8.10.1 Mapping tab

Use the Mapping tab to specify how the selected output column is derived (or mapped). You can specify any valid expression.

Most commonly, mapping expressions contain table columns and functions.

- Enter input column names or drag columns from the input schema and drop them in the box on the Mapping tab.
- Insert functions by entering them directly, using the smart editor, or by clicking the Functions button to open the function wizard.

After you map your source to the XML_Map transform, you might determine that you need to use another transform before you send the data to the XML_Map transform. For example, you might add a validation transform to ensure that only data with a certain format is passed or you might add a Case transform to send only a subset of the data.

In general, when you change an input schema to the XML_Map transform, the Designer checks the existing top-level mappings to determine if any remapping is required.

- If the mapping contains a column with a table name that is not a current input schema name and the column is in the new input schema, the Designer automatically replaces the table name with the new input schema name. Specifically, the Designer automatically updates the input schema name for each matching column in the following mapping configuration tabs of the XML_Map editor:
  - Mapping
  - Iteration Rule
  - WHERE
  - DISTINCT
  - ORDER BY
  - GROUP BY
- If the mapping contains a column that was in the obsolete table, but the column does not exist in the new input schema, you must either remove the column or remap it from the original source.

The Designer does not automatically remap the input schema for the following situations:

- When you connect a new source to the XML_Map transform before you disconnect the old source. You must click the *Schema Remapping* button on the Mapping tab to update the input schema name for columns in the Mapping, WHERE, GROUP BY, and ORDER BY tabs.
- When the source is a nested schema and you either change the source to a similar nested schema, or you add or delete a transform before the XML_Map transform. Click the *Schema Remapping* button to update the Mapping input schema name.

Schema to schema mapping

You can map a source schema to a target schema by making the target schema current and entering the source schema path in the Mapping tab. In this case, the software assumes the source and target schemas have the same structure, including the number, order and data type of columns in each level of the structure.

When you specify schema to schema mapping, you cannot also have an iteration rule, but you may specify columns in the DISTINCT, WHERE, GROUP BY, and ORDER BY tabs.

Merge To operation

When you make a target schema current in the Mapping tab, the *Merge To* operation becomes available. The Merge To operation allows you to copy and paste a target schema at the same level, create mappings to different sources for the copied target schema and the original schema, and then merge the result sets. In any transforms that follow the XML_Map transform, only the original target schema is displayed.

3.5.5.8.10.1.1 To remap when automatic remapping was not done in the XML_Map transform

1. In the Mapping tab, click the *Schema Remapping* button. The *Replace Obsolete Schema window* opens.
2. In the *Specify obsolete schema* drop-down list, choose the source schema that you disconnected from the XML_Map transform.
   This list displays only the top-level input schema. For an obsolete nested schema, enter the name of the top-level schema.
3. In the *Choose correct schema* list, choose the output schema of the transform that you added between the source and XML_Map transform.
4. Click *Remap*.
   A message displays the number of columns that were remapped; for example:

```
Schema "ODS_SALESORDER" was replaced by schema "Validation_Pass" in 11 column
names.
```

3.5.5.8.10.2 Iteration Rule tab

Use the Iteration Rule tab to define how the output data set for the selected output schema is calculated. An iteration rule is associated only with a repeatable target node, and defines how to construct the instances of the target schema from the source data. It is a mechanism to specify the input data sets and the way the software should join them to create the target data set.

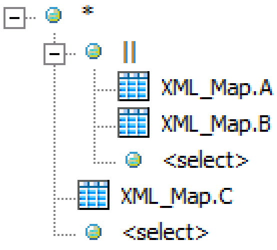
Data Services supports different kinds of joins in the iteration rule: INNER JOIN, LEFT OUTER JOIN, CROSS JOIN, and PARALLEL JOIN. PARALLEL JOIN is not a standard SQL join.

In the iteration rule tab, a hierachical tree represents a logical combination of operations and input schemas that form a rule. Each operation in the rule is displayed as a node and may contain other operations or input schemas as children.

For example, a rule that performs a parallel operation on example tables A and B, and then combines that output set with table C by using a cross operation might logically look like this:

```
(A || B) * C
```

In the iteration rule tab, this same rule might look like this:



Constructing iteration rules

Use the iteration rule tab to create iteration rules for each repeatable schema in your output:

- To add a new element to the rule, click the *<select>* placeholder under an operation node and choose the new operation or input schema from the drop-down list.
  Elements that can be added to an iteration rule include the following:
  - *INNER JOIN*
    Performs a SQL INNER JOIN on the sources. Create the expression to use for the join condition in the On area of the rule editor.
    When you create the expression, you can use the following types of columns:
    - Source columns from the sources under the current operation and the left side of the current iteration rule tree.
    - Source columns from the sources that appear in the iteration rules associated with the parent schemas of the current target schema.
    - Target columns from the parent schemas of the current target schema.

    When using a source column, the path from the column being used to the source schema must contain no repeatable schemas.
    When using a target column, it must be a scalar column and descend from the parent schema of the schema where the iteration rule is defined. In addition, the path from the parent schema to the target column must contain no repeatable schemas.
  - *LEFT OUTER JOIN*
    Performs a SQL LEFT OUTER JOIN on the sources. Create the expression to use for the join in the On area of the rule editor.
    When you create the expression, you can use the following types of columns:
    - Source columns from the sources under the current operation and the left side of the current iteration rule tree.
    - Source columns from the sources that appear in the iteration rules associated with the parent schemas of the current target schema.
    - Target columns from the parent schemas of the current target schema.

    When using a source column, the path from the column being used to the source schema must contain no repeatable schemas.
    When using a target column, it must be a scalar column and descend from the parent schema of the schema where the iteration rule is defined. In addition, the path from the parent schema to the target column must contain no repeatable schemas.

- ○ ** - Cross operation*
  Produces a Cartesian product of two or more sources. When the sources have no parent-child relationship, the behavior is the same as a standard SQL CROSS JOIN. When the sources have a parent-child relationship, the Cartesian operation provides a mechanism to iterate through all instances of the repeatable elements identified by the source schemas in the operation in the document order.
- ○ || - Parallel operation
  The Parallel operation is not a standard SQL operation. It takes two or more sources and combines corresponding rows in each source to generate the output set. For example, the first rows in a pair of input tables is combined to become the first row of the output set, the second rows are combined to become the second output row, and so on.
  If the sources have different numbers of rows, the output set will contain the same number of rows as the largest source. For extra rows in the output set that contain data from only one source, the additional columns that would contain data from the other sources are considered empty.
- ○ Available input schemas

- To remove an element from the rule, click the element and choose *<delete>* from the drop-down list.
  If you remove an operation node, any child operations or schemas will also be removed from the rule.
- To change an operation type, click the operation node and choose the new operation from the drop-down list.

i Note

There is no limit to the number of sources that may be used in an iteration rule.

Automatic rule generation

The iteration rule can be generated automatically. After you have created mappings for the columns under the current target schema, click *Propose rule* in the Iteration Rule tab. The software generates the iteration rule tree. Always validate that the generated iteration rule matches your requirements. Modify the rule as needed, and add the ON condition expression when appropriate.

You can also propose rules recursively. When you click *Propose rule recursively*, the software recursively moves through the target tree under the current target schema, finds all repeatable schemas, and generates the iteration rule for each repeatable schema based on the mappings under the schema.

i Note

Automatic rule generation is a best-guess function. For example, the software cannot know the ON condition, or whether to use INNER JOIN or LEFT OUTER JOIN. Use the automatic rule generation as a guide and always verify that the iteration rule that it creates fits your needs.

3.5.5.8.10.3 WHERE tab

A WHERE clause can be created for any target schema in the output structure. Use the WHERE tab to set conditions that determine which rows are output. Enter the conditions in SQL syntax, as you would a WHERE clause in a SQL SELECT statement. The WHERE tab applies to the current output schema.

You can specify any valid expression. To enter conditions, do one of the following:

- Enter expressions in the editor.
- Drag columns from the input or output schema area to the editor.
- Use the Functions button. Use the pushdown_sql function to have Data Services create WHERE clauses dynamically based on data rather than pre-specifying the clause.

  Note

  The pushdown_sql fuction can be used if the immediate input to the XML_Map transform is the table source where you want to push the WHERE clause.

Source and target columns may be used in the WHERE expression.

Restriction

Source columns must come from the source schemas in the current iteration rule or those that appear in the iteration rules associated with the parent schemas of the current target schema. Additionally, the path from the column being used to the source schema must contain no repeatable schemas.

Target columns must come from the current target schema or parent schemas of the current target schema. Additionally, the path from the column being used to the target schema must contain no repeatable schemas.

Note

If your expression contains varchar comparisons, Data Services ignores trailing blanks in the data. For Oracle data, use the rtrim or rpad functions if the number of trailing blanks might differ on either side of the comparison.

3.5.5.8.10.4 DISTINCT tab

Use the DISTINCT tab to specify the input or output schema columns that should be used to determine whether a row is distinct. If the column specified in the DISTINCT tab contains a distinct value, the row is a new output row. The DISTINCT tab applies to the current output schema.

To add a column to the Distinct columns list, select the column in the output schema area and drag it to the box in the *DISTINCT* tab. The Designer adds the column to the bottom of the list.

To remove a column, use one of the following options:

- Right-click the column and select *Delete*.
- Select the column and click the delete icon in the top right corner of the DISTINCT tab.

To consider the entire output row as distinct, select the *Whole row is DISTINCT* option.

Note

You cannot specify both source and target columns at the same time in the DISTINCT tab.

When source columns are used, they must descend from the source schemas in the current iteration rule. In addition, the path from the source schema to the column must contain no repeatable nodes.

When target columns are used, they must descend from the current target schema. In addition, the path from the current target schema to the column must contain no repeatable nodes.

3.5.5.8.10.5 GROUP BY tab

Use the GROUP BY tab to specify a list of columns for which you want to combine output. For each unique set of values in the group by list, Data Services combines or aggregates the values in the remaining columns. For example, you might want to group sales order records by order date to find the total sales ordered on a particular date. The GROUP BY tab applies to the current output schema.

To add a column to the Group By list, select the column in the input or output schema area and drag it to the box in the *GROUP BY* tab. The Designer adds the column to the bottom of the list.

The first column listed is used for primary grouping, the second column listed is used for secondary grouping, and so forth. To change the groupings, use one of the following options:

- Right-click the column and select *Move Up* or *Move Down*.
- Select the column and click the down or up arrow in the top right corner of the GROUP BY tab.

> Note
>
> You can specify either source or target columns in the GROUP BY column list.
>
> When source columns are used, they must descend from the source schemas in the current iteration rule. In addtion, the path from the source schema to the column must contain no repeatable nodes.
>
> When target columns are used, they must descend from the current target schema. In addition, the path from the current target schema to the column must contain no repeatable nodes.

To remove a column, use one of the following options:

- Right-click the column and select *Delete*.
- Select the column and click the delete icon in the top right corner of the GROUP BY tab.

If you specify a group by list, then all columns in the output schema must be either in the group by list or mapped to an aggregate function, such as avg, count, max, min, or sum.

Grouping methods

While the GROUP BY operation is similar to the standard SQL GROUP BY operation, it does not always operate in exactly the same way. The XML_Map transform groups output items in different ways depending upon the columns specified and whether or not aggregation functions are used:

- Simple grouping
  The XML_Map transform groups output items together according to the unique values of the GROUP BY list when the following conditions are met:
  - Source or target columns are specified in the GROUP BY list.
  - If source columns are specified, no aggregation functions are defined in the target schema.

  In this grouping method, the operation does not remove any items from the output data set.

- Group aggregation
  The XML_Map transform performs grouping exactly like a standard SQL GROUP BY clause when the following conditions are met:
  - Source columns are specified in the GROUP BY list.
  - Aggregation functions are defined under the current target schema.
  - Columns in the aggregation functions descend from the sources in the current iteration rule.
  - Paths from the iterating sources to the columns do not contain any repeatable nodes.

  > Note
  >
  > All columns in the output schema must be either part of the group by list or mapped to an aggregate function such as avg, count, max, min, or sum.
- Instance aggregation
  The XML_Map transform evaluates the aggregation functions for each of the items in the output data set when the following conditions are met:
  - Aggregation functions are defined under the current target schema.
  - Columns used in the aggregation functions descend from the sources in the current iteration rule.
  - Paths from the sources to the columns being used contain repeatable nodes.

  The XML_Map transform also evaluates the aggregation functions for each of the items in the output data set when the following conditions are met:
  - Aggregation functions are defined under the current target schema.
  - Columns used in the aggregation functions descend from the current target schema.
  - Paths from the current target schema to the columns being used contain repeatable nodes.

> ⚠ Restriction
>
> You cannot use both group and instance aggregation at the same time.

3.5.5.8.10.6 ORDER BY tab

Use the ORDER BY tab to specify the columns you want used to sort the output data set. The ORDER BY tab applies to the current output schema.

To add a column, select the column in the input or output schema area and drag it to the box on the ORDER BY tab. The Designer adds the column to the bottom of the list.

The first column listed is used for primary sorting, the second column listed is used for secondary sorting, and so forth. To change the column order, use one of the following options:

- Right-click the column and select *Move Up* or *Move Down*.
- Select the column and click the down or up arrow in the top right corner of the *ORDER BY* tab.

You can specify either source or target columns in the ORDER BY tab.

When source columns are used, they must descend from the source schemas in the current iteration rule. In addition, the path from the source schemas to the column must contain no repeatable nodes.

When target columns are used, they must descend from the current target schema. In addition, the path from the current target schema to the column must contain no repeatable nodes.

To remove a column, use one of the following options:

- Right-click the column and select *Delete*.
- Select the column and click the delete icon in the top right corner of the *ORDER BY* tab.

The default sort order is ascending. To change the order, select *Ascending* or *Descending* from the adjacent drop down box.

3.5.5.8.10.7 Advanced tab

Use the options in the Advanced tab to run the XML_Map transform in a separate process, or to configure options for the input schema(s). The options in the Advanced tab apply to the entire XML_Map transform.

Use the Input schemas table to configure additional options for the input schema(s) used in the transform:

Option	Description
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. Available values: • *Automatic* (default) • *Yes* • *No* Cache specified in the Advanced tab of the XML_Map transform editor overrides any cache specified in a source.

3.5.5.8.10.8 Find tab

Use the Find tab to search for a specific word or term in the input schema or the output schema.

Related Information

Searching in an input or output schema [page 1485]

3.5.5.8.11 Example: Nesting data with the XML_Map transform

In this example, assume you have a database table containing a list of company employees and department information. You want to create a structure that has a list of departments, each containing a list of employees and a new column that contains the number of employees in the department.

Source	Target

`Employee`
` |-departmentID`
` |-departmentName`
` |-employeeName`
` |-employedDate`
	`Company`
	` |-department (*)`
	`    |-departmentID`
	`    |-departmentName`
	`    |-employee (*)`
	`       |-employeeName`
	`       |-employedDate`
	`    |-totalEmployees`

1. Create iteration rules for the `department` and `employee` schemas.

   Both schemas require information from the `Employee` source table, so they must iterate on that; no JOINs or other operators are required.
2. Create a new department instance for each individual department, based on the department ID.

   Include the `Company.department.departmentID` target column in the DISTINCT tab for the `department` schema. Each time a new value in that column is encountered, a new `department` instance will be created.
3. Identify and create an `employee` instance for each employee that belongs to the department.

   Because the ID for the current department is already known, you can use it in an expression in the WHERE tab for the `employee` schema to include only the correct employees:

   ```
   Employee.departmentID = Company.department.departmentID
   ```

   When the `employee` instance iterates against the source, only rows that have the matching department ID will be selected.
4. Aggregate the number of employees in the department.

   The `employee` instances have already been created, so you can use those to create a mapping expression for the `totalEmployees` column:

   ```
   count(Company.department.employee)
   ```

3.5.5.8.12 Example: Unnesting data with the XML_Map transform

In this example, assume that you have an XML structure that contains information about purchase orders. You want to generate a flat list for the items in all of the purchase orders, ordered by the total sales for each item.

<table>
<tr><th>Source</th><th>Target</th></tr>
<tr><td><pre>
purchaseOrders
    |-purchaseOrder (*)
    |-sellerParty
       |-sellerID
       |-sellerName
    |-buyerParty
       |-buyerID
       |-buyerName
    |-orderLine
       |-item (*)
          |-name
          |-quantity
          |-unitPrice
          |-currency
</pre></td><td><pre>
item
 |-sellerID
 |-buyerID
 |-itemName
 |-totalSales
 |-currency
</pre></td></tr>
</table>

1. Create an iteration rule for the `item` output schema.

   Because the output schema requires data from columns in multiple nested schemas, use a cross (*) operation to flatten the data.

   ```
   *
   |-purchaseOrders.purchaseOrder
   |-purchaseOrders.purchaseOrder.orderLine.item
   ```

   The input ports are always assumed to be repeatable, which means that the software expects that multiple documents of `puchaseOrders` may come in.
2. Sort the output set by the total sales for each item.
   a) Calculate the total sales amount for each item.

      Use information from the source columns in an expression in the mapping for the `totalSales` column:

      ```
      orderLine.item.quantity * orderLine.item.unitPrice
      ```
   b) Order the output set.

      Include the `item.totalSales` target column in the ORDER BY tab for the `item` output schema.

3.5.5.8.13 Example: Transforming a hierarchical source to a different hierarchical target

In this example, assume that you have a hierarchical structure that contains a catalog of books. The catalog includes information associated with each book, such as name, price, quantity sold, and information about the author. You want to transform this data into a structure that is instead organized by author. You also want to calculate the total sales for each book.

<table>
<tr><th>Source</th><th>Target</th></tr>
<tr><td><pre>
catalog
 |-book
    |-Name
    |-Price
    |-Quantity
</pre></td><td><pre>
authors
 |-Author
    |-Name
       |-firstName
       |-lastName
    |-Address
</pre></td></tr>
</table>

Source	Target
|-Author |-firstName |-lastName |-Author_nt_1 |-street |-city |-state |-zip	|-Author_nt_1 |-street |-city |-state |-zip |-book |-Name |-Price |-Quantity |-totalSales

1. Create an iteration rule for the `Author` output schema.

   Because the output schema requires data from columns in multiple nested schemas, use a cross (*) operation to flatten the data.

   ```
   *
   |-catalog.book
   |-catalog.book.Author
   ```

2. Create a new Author instance for each individual author, based on the first and last name of the author.

   Include the `authors.Author.Name.firstName` and `authors.Author.Name.lastName` target columns in the DISTINCT tab for the `Author` target schema. Each time a new combination of the values in those columns is encountered, a new `Author` instance will be created.

3. Sort the Author instances by the name of each author.

   Include the `authors.Author.Name.firstName` and `authors.Author.Name.lastName` target columns in the ORDER BY tab for the `Author` target schema.

4. Because there may be more than one line for the street portion of the author's address, create an iteration rule for the `Author_nt_1` output schema.

   ```
   catalog.book.Author.Author_nt_1
   ```

5. Map the author information from the source schema to the target schema.

   Include the appropriate source column or schema in the Mapping tab for each output column or schema:

Target column or schema	Mapping expression
authors.Author.Name.firstName	`catalog.book.Author.firstName`
authors.Author.Name.lastName	`catalog.book.Author.lastName`
authors.Author.Address.Author_nt_1	`catalog.book.Author.Author_nt_1`
authors.Author.Address.Author_nt_1.street	`catalog.book.Author.Author_nt_1.street`
authors.Author.Address.city	`catalog.book.Author.city`
authors.Author.Address.state	`catalog.book.Author.state`
authors.Author.Address.zip	`catalog.book.Author.zip`

6. Create an iteration rule for the `book` output schema.

Like the `Author` output schema, the `book` output schema requires a combination of the `catalog.book` and `catalog.book.Author` source schemas. Use a cross (*) operation to flatten the data.

```
*
|-catalog.book
|-catalog.book.Author
```

7. Identify and create a `book` instance for each book that belongs to the author.

   Because the first and last names of the author of the book are already known, you can use them in an expression in the WHERE tab for the `book` schema to include only the correct books:

   ```
   catalog.book.Author.firstName = authors.Author.Name.firstName AND
   catalog.book.Author.lastName = authors.Author.Name.lastName
   ```

8. Map book information from the source schema to the target schema.

   Include the appropriate source column in the Mapping tab for each output column:

Target column	Mapping expression
authors.Author.book.Name	`catalog.book.Name`
authors.Author.book.Price	`catalog.book.Price`
authors.Author.book.Quantity	`catalog.book.Quantity`

9. Calculate the total sales for each book.

   Use information from the source columns in an expression in the mapping for the totalSales column:

   ```
   catalog.book.Price * catalog.book.Quantity
   ```

3.5.6 Text Data Processing transforms

3.5.6.1 Entity Extraction transform

The Entity Extraction transform performs linguistic processing on content by using semantic and syntactic knowledge of words. You can configure the transform to identify paragraphs, sentences, and clauses and it can extract entities and facts from text. Typically, you use the Entity Extraction transform when you have text with specific information you want to extract and then use in downstream analytics and applications.

3.5.6.1.1 Extraction options

The Entity Extraction transform includes options that control which language, dictionaries, and rules to use for extraction. The *Processing Options* group includes specific configuration parameters for processing.

3.5.6.1.1.1 Common

The *Common* option group includes a setting to run the transform as a separate process.

Option	Description
Run as Separate Process	*Yes*: Splits the transform into a separate process. *No*: Keeps the transform as a single, possibly multithreaded, process.

3.5.6.1.1.2 Languages

The *Languages* option group includes settings to process content in different languages, such as English, German, and French. If the input content is in a language other than the specified languages, you might see unexpected results.

Option	Description
Language	Specifies the language for processing your content. You may select another language from the list of available languages displayed alphabetically in the drop-down menu. • The default language setting is 'Auto'. You may select another language from the list, to override the automatic identification. If the transform cannot identify the language, it uses the setting of the *Default Language* option. Note If content arrives in XML, and is organized in sections, the tranform identifies the language of each section. • If you select a language other than 'Auto', only entities defined for that language will be extracted for the entire document. When the language is set to 'Auto', any specified dictionaries or rule files the name of which do not identify a language will always be applied. However, dictionaries and rule files that do identify a language (e.g. English) in the file name will be applied only to input identified in that language. Note You will not be able to run extraction unless you have a language directory that contains the files for at least one language. By default, the language directory is installed on a client as well as a server during installation. The default location for the language directory is: `<LINK_DIR>/TextAnalysis/languages`.
Default Language	Specifies the default language that the transform should assume if the Language option was *Auto* and the transform could not identify a language. If the value 'None' is chosen as the value for *Default Language*, and the language cannot be identified, a non-fatal error occurs and processing continues.

Option	Description
	If the name of some other available language is chosen as the default language, and the transform fails to identify a language on the first attempt, only entities defined for the chosen language will be extracted.
Filter By Entity Types	Specifies a list of entity types (supported by the selected language) to use for filtering the extraction output. Note When the Language = 'Auto', this list is the union of all of the available entity types from the installed languages. By default, a drop-down menu showing '...' displays. Clicking this launches the *Ordered Options Window - [Filter By Entity Types - Option]* dialog. Select or remove one or more entity types from the list of available entity types for that language. Note Entity type support varies among languages.

3.5.6.1.1.3 Processing Options

The *Processing Options* group includes configuration settings for the transform. They affect how the transform will process the content before generating the extraction output.

The *Dictionary Only* option is most useful when you want to extract entities based solely on entities defined in a dictionary. For example, you want to match exactly the product and customer names from your custom dictionary and you are not interested in any other extraction output. In such a case, getting predefined entities from the extraction process will not be of interest.

> Note
>
> Predefined entities are entities associated with different languages and are part of the language modules. These entities are extracted by default.

The *Processing Timeout* option is most useful when you want to limit the amount of time spent on processing large content or content that take a very long time to process.

Option	Description
Dictionary Only	Use this option to limit the extraction process to use entities defined only in the specified dictionaries. You must specify a dictionary file to use this option.

Option	Description
	Note If you select this option, the extraction output will not include any predefined entities. Along with this option, if you also select the *Rule* option, the extraction output will include entities and facts defined in the rules along with entities from the specified dictionaries.
Advanced Parsing	Specifies whether advanced parsing information should be produced during extraction. Advanced parsing enriches linguistic processing including richer noun phrase structure, noun phrase coordination, and syntactic function attributes that can be leveraged in custom rules. This option is available only for the English language. By default, *YES* and *NO* display. If you select the *YES* option for non-English languages, an error message displays.
Processing Timeout	Use this option to stop processing the content after a set amount of time. By default, the *Processing Timeout* option is set to 60 seconds. The Processing Timeout value can be one of the following: `-1` indicates no timeout should be enforced. `>=1` indicates the amount of time (in seconds) after which processing should abort.
Document Properties	Specifies whether document properties of a binary document should be extracted, if they are present in the document. A value of *YES* causes the extraction, and a value of *NO* (the default) causes no extraction. Document properties are name-value pairs. The Entity Extraction transform extracts only the following document properties for binary documents: • APP_NAME: The name of the software that was used to create the document • APP_VERSION: The version of the software that was used to create the document • AUTHOR: The name of the person who created the document • COMPANY: The name of the company that owns the document • TITLE: The title of the document • DATE_CREATED: The date on which the document was created Document properties, if available, are extracted as entities. The SOURCE for the properties is called DOC_PROPERTY and only the following fields are defined for DOC_PROPERTY entities: • ID: The entity ID of the document property • SOURCE: DOC_PROPERTY • TYPE: The name of the document property • STANDARD_FORM or SOURCE_FORM: The value of the document property • CONVERTED_TEXT: The textual content of the binary document Any other output columns are not applicable to DOC_PROPERTY extraction rows, and have their value set to `-1`.

3.5.6.1.1.4 Dictionaries

The *Dictionaries* option group includes settings to process content by specifying one or more dictionaries that should be used when performing extraction. It also enables filtering by entity types defined in each dictionary.

The *Dictionaries* option group is comprised of individual dictionaries. You can configure the transform to use multiple dictionaries. These options are found under ▶ *Dictionaries* ▶ *Dictionary* ▶ *Dictionary File* ◀.

Option	Description
Dictionary	Use this option to add dictionaries that should be used during extraction or delete an existing dictionary. Right-click this option and select the option to duplicate an entry or to delete an entry. Once the entry is duplicated, change the duplicate dictionary file by selecting the dictionary to use from the directory structure.
Dictionary file	Use the *Browse* option under the drop-down menu to select a valid, compiled dictionary file to use for extraction. ℹ Note To include the dictionaries during extraction, they need to be accessible to the job server. If the dictionary files are located on a remote computer, include the path to those files (that can be resolved by the job server).
Filter By Entity Types	Specifies a list of entity types (defined in the selected dictionary) to use for filtering the extraction output. By default, a drop-down menu showing '...' displays. Clicking this launches the *Ordered Options Window - [Filter By Entity Types - Option]* dialog. Select or remove one or more entity types from the available entity list.

Related Information

Designer Guide: Dictionary overview [page 320]
Designer Guide: Executing Jobs, Changing Job Server options, To use mapped drive names in a path [page 432]

3.5.6.1.1.5 Rules

The *Rules* option group includes settings to process content by specifying one or more extraction rules to use when performing extraction. It also enables filtering by rule names defined in each rule file.

The *Rule* option group includes individual rules. You can configure the transform to use multiple rules. These options are found under ▶ *Rules* ▶ *Rule* ▶ *Rule File* ◀.

Option	Description
Rule	Use this option if you want to add rules that should be used during extraction or to delete an existing rule. Right-click on this option and select the option to duplicate an entry or to delete an entry. Once the entry is duplicated, change the duplicate rule file by selecting the rule you want to use from the directory structure.
Rule File	Use the *Browse* option under the drop-down menu to select a valid, compiled rule file to use for extraction. To include the rules during extraction, they need to be accessible to the job server. If the rule files are located on a remote computer, include the path to those files (that can be resolved by the job server). Note A rule file typically contains multiple rules. You can use the rule filtering option to select a specific rule in a rule file.
Filter By Rule Names	Specifies a list of rule names (defined in the selected rule file) to use for filtering the extraction output. By default, a drop-down menu showing '...' displays. Clicking this launches the *Ordered Options Window - [Filter By Rule Names - Option]* dialog. Select or remove one or more rules from the filtering list.

Related Information

Designer Guide: Rule Overview [page 320]

Designer Guide: Executing Jobs, Changing Job Server options, To use mapped drive names in a path [page 432]

3.5.6.1.2 Input fields

The following is a Data Services recognized input field that you can use in the input mapping for the Entity Extraction transform.

Name	Data Type	Description
TEXT	Long, blob, or varchar	Mandatory field. It includes content to be processed by the transform to extract entities and/or facts. The content must be in a text format such as a text, HTML, XML, or certain binary-formats (such as PDF).
TEXT_ID	Long, int, or varchar	Optional field. Unique identifier to be used for tracing the content in case of an error.

Name	Data Type	Description
		Note - An unsupported data type is ignored during runtime and instead either the file name (if read from an unstructured text file format) or the string `TEXT` input field is be used as the content identifier. - When a `varchar` or `long` column is mapped to the `TEXT_ID` input field and a value used to construct an error message contains more than 1K bytes, the value will be truncated to 1K.

3.5.6.1.3 Output fields

The following are Data Services recognized output fields that you can use in the output mapping for the Entity Extraction transform. The fields are listed in the order they appear on the *Output* tab.

Generated field name	Data Type	Description
ID	int	Represents a parent-child relationship between entities and/or facts. This value is unique within the scope of the processed input text. Note If you process two different input documents using the same data flow and store the output to a database, you should not use this field as a primary key.
PARENT_ID	int	Represents a parent-child relationship between entities and/or facts. If present, it provides a link to a parent ID value. If not present, this value is set to -1 to indicate there is no relationship.
STANDARD_FORM	varchar (2000)	The standard form of an entity, fact, or subfact. Generally it is the longest, most precise or official name associated with the value of the corresponding `TYPE` column. Note The standard form and the source form for an entity are often the same.
TYPE	varchar (255)	The type of an entity or fact. It may also represent subtypes or subfact types if applicable. For example, “Mr. Jones” will be identified as a `PERSON` entity and “car” as a `COMMON_VEHICLE/LAND` entity subtype.

Generated field name	Data Type	Description
		Note "/" is used as a separator to identify subtypes.
SOURCE_FORM	varchar (2000)	The name of an entity, fact, or subfact as mentioned in the input text.
SOURCE	varchar (10)	The origin of an entity or fact. Meaning, how the match was determined, based on one of the following: • SYSTEM - indicating that the entity was matched using the system files. • DICTIONARY - indicating that the entity was matched using a dictionary. • RULE - indicating that the entity or fact was matched using an extraction rule file.
OFFSET	int	The character offset of an entity or a fact in the CONVERTED_TEXT field.
LENGTH	int	The character length of an entity or a fact in the CONVERTED_TEXT field.
PARAGRAPH_ID	int	A unique identifier of the paragraph in the CONVERTED_TEXT field containing the entity or fact.
SENTENCE_ID	int	A unique identifier of the sentence in the CONVERTED_TEXT field containing the entity or fact.
CONVERTED_TEXT	long	The content text representation in UTF-16 encoding of the input text. Note When the CONVERTED_TEXT output column is selected, the first entity/fact output row for any input document will contain the UTF-16 textual representation of the input document. Any of the subsequent entity/fact output rows for the input document will not contain textual representation of the input document.

Example

Extraction text and output fields values

The following table shows partial results of the extraction output you may see when the following sample text is processed by the Entity Extraction transform. It shows values for the following fields:

- ID
- PARENT_ID
- STANDARD_FORM
- TYPE
- SOURCE_FORM

- SOURCE

Sample input text: "Mr. Jones is very upset with Green Insurance. The offer for his totaled vehicle is too low. He states that Green offered him $1250.00 but his car is worth anywhere from $2500 to $4500. Mr. Jones would like Green's comprehensive coverage to be in line with other competitors."

ID	PARENT_ID	STANDARD_FORM	TYPE	SOURCE_FORM	SOURCE
1	-1	Mr. Jones	PERSON	Mr. Jones	SYSTEM
2	-1	Mr. Jones is very upset with Green Insurance.	Sentiment	Mr. Jones is very upset with Green Insurance.	RULE
3	2	Mr. Jones	Topic	Mr. Jones	RULE
4	2	very upset	StrongNegative-Sentiment	very upset	RULE
5	2	Green Insurance	Topic	Green Insurance	RULE
6	-1	Green Insurance	PROP_MISC	Green Insurance	SYSTEM
7	-1	The offer for his totaled vehicle is too low.	Sentiment	The offer for his totaled vehicle is too low.	RULE
8	7	totaled vehicle	Topic	totaled vehicle	RULE
9	7	too low	MinorProblem	too low	RULE
10	-1	totaled vehicle	COMMON_VEHICLE/OTHER	totaled vehicle	SYSTEM
11	-1	He states that Green offered him $1250.00 but his car is worth anywhere from $2500 to $4500.	Sentiment	He states that Green offered him $1250.00 but his car is worth anywhere from $2500 to $4500.	RULE
12	11	car	Topic	car	RULE
13	11	worth anywhere from $2500 to $4500	WeakPositiveSentiment	worth anywhere from $2500 to $4500	RULE
14	-1	Green	PROP_MISC	Green	SYSTEM
15	-1	$1250.00	CURRENCY	$1250.00	SYSTEM
16	-1	car	COMMON_VEHICLE/LAND	car	SYSTEM
17	-1	from $2500 to $4500	CURRENCY	from $2500 to $4500	SYSTEM
18	-1	Mr. Jones	PERSON	Mr. Jones	SYSTEM

ID	PARENT_ID	STANDARD_FORM	TYPE	SOURCE_FORM	SOURCE
19	-1	Mr. Jones would like Green's comprehensive coverage to be in line with other competitors.	Request	Mr. Jones would like Green's comprehensive coverage to be in line with other competitors.	RULE
20	19	Mr. Jones	Topic	Mr. Jones	RULE
21	19	would like	GeneralRequest	would like	RULE
22	19	Green's comprehensive coverage	Topic	Green's comprehensive coverage	RULE
23	-1	Green	PROP_MISC	Green	SYSTEM
24	-1	other competitors	COMMON_PERSON /GROUP	other competitors	SYSTEM
25	-1	other competitors	COMMON_ ORGANIZATION/ COMMERCIAL	other competitors	SYSTEM

In the above example, row 1 shows ID as 1 (unique identifier) with PARENT_ID as -1 (no parent relationship). The TYPE column shows entity (PERSON), facts (Sentiment), and subfacts (StrongNegativeSentiment). To view the content of the CONVERTED_TEXT field in the Designer, use the `long_to_varchar` function to perform the conversion.

Related Information

Designer Guide: XML extraction and parsing for columns [page 366]

3.6 Functions and Procedures

In Data Services, functions take input values and produce a return value. Procedures take input values and perform a set of operations without returning a specific value. Input values can be parameters passed into a data flow, values from a column of data, or variables defined inside a script. This section discusses functions and procedures, including detailed descriptions of built-in functions—the input parameters and required syntax, and the return values and data types.

3.6.1 About functions

3.6.1.1 Functions compared with transforms

Some functions can produce the same or similar values as transforms. However, functions and transforms operate in a different scope:

- Functions operate on single values, such as values in specific columns in a data set.
- Transforms operate on data sets, creating, updating, and deleting rows of data.

SAP Data Services does not support functions that include tables as input or output parameters, except functions imported from SAP Applications.

3.6.1.2 Operation of a function

The function's operation determines where you can call the function. For example, the Lookup database function operates as an iterative function. The lookup function can cache information about the table and columns it is operating on between function calls. By contrast, conversion functions, such as to_char, operate as stateless functions. Conversion functions operate independently in each iteration. An aggregate function, such as max, requires a set of values to operate. Neither the lookup function (iterative) nor the max function (aggregate) can be called from a script or conditional where the context does not support how these functions operate.

The function type determines where a function can be used:

Type	Description
Aggregate	Generates a single value from a set of values. Aggregate functions, such as max, min, and count, use the data set specified by the expression in the *Group By* tab of a query. Can be called only from within a Query transform—not from custom functions or scripts.
Iterative	Maintains state information from one invocation to another. The life of an iterative function's state information is the execution life of the query in which they are included. The lookup function is an iterative function. Can be called only from within a Query transform—not from functions or scripts.
Stateless	State information is not maintained from one invocation to the next. Stateless functions such as to_char or month can be used anywhere expressions are allowed.

3.6.1.3 Arithmetic in date functions

Data Services performs some implicit data type conversions on date, time, datetime, and interval values.

Related Information

Date arithmetic [page 1039]

3.6.1.4 Including functions in expressions

You can use functions in the following:

- Transforms (Query , Case, SQL, Map_Operation)
- Script objects
- Conditionals
- Other custom functions

Before you use a function, you need to know if the function operation makes sense in the expression you are creating.

For example:

- The `max` function cannot be used in a script or conditional where there is no collection of values on which to operate.
- Parameters can be output by a work flow but not by a data flow.

You can use two editors to add an existing function to an expression. These are:

- Smart editor
  Embedded in other editor windows like the Script Editor, Conditional Editor, and Query Editor, the smart editor offers color coded syntax, a right-click menu, keyboard short cuts, and a list of available variables, data type formats, and functions that you can use to define a function.
- Function wizard
  You can use the function wizard to define the parameters for an existing function. The function wizard offers the most help when defining complex functions.

Related Information

Smart editor [page 1021]

3.6.1.4.1 To create an expression that includes an existing function

1. Go to the script, query, or conditional editor in which you will include the expression.
2. Enable the smart editor and begin entering your expression.
3. When you want to include the function, click the *Functions* button.

   The Designer opens the Select Function window of the function wizard.

4. Select a category in the *Function categories* box.

   A list of functions in that category appears in the Function name box. The functions shown depend on the object you are using. For example, the functions available for ABAP data flows are a subset of those available for data flows.

   In some cases, it does not make sense to use a function even though it is available. For example, the SQL function can be called in a mapping expression or a WHERE clause, but it would result in a SQL statement inside the SQL statement generated to execute a data flow.
5. Select a specific function in the *Function name* list.

   A description of the function appears below the boxes.
6. Click *Next*.
7. Enter the values required by the function in the text boxes.

   The page shown for each function is unique. Each page is designed to help you construct the current function. The example below shows the most common layout, however more complex functions may use different layouts.

   Click in a box to see a description of the parameter at the bottom of the window. Use the down-arrow button to select input parameters. Use the smart editor button (...) to see a larger input box.
8. Click *Finish*.

   The function and the parameters appear in the smart editor.

3.6.1.4.2 To edit an existing function call in an expression

You can edit function calls from a variety of editors. For example, you can edit them from the smart editor, script editor and so on. Using the function wizard to edit your complex function calls may save some time as the different pieces of the function are parsed into separate arguments. Then, you can identify the options you need to change.

> Note
>
> If your function contains specific spacing, line breaks or comments, you may not want to invoke the function wizard. Using the function wizard to edit an existing function call removes your formatting, spacing and comments. After you finish editing in the function wizard, the function text appears on one line with minimal spacing between the parameters.

1. Go to the expression that contains the function call that you want to change.
2. Right-click on the text of the function call and select *Edit Function*.
   The function wizard opens.
3. Make changes to the function call, and then click *Finish*.
   The function wizard closes and you can see the results of your function call.

Related Information

Operation of a function [page 1512]

Including functions in expressions [page 1513]

3.6.2 Built-in functions

SAP Data Services provides a set of built-in functions.

In the software, database and application functions, custom functions, and most built-in functions can be executed in parallel within the transforms in which they are used.

You can run each resource-intensive functions, such as lookup_ext and count_distinct, as a separate sub data flow that uses separate resources (memory and computer) from each other.

Related Information

Performance Optimization Guide: Degree of parallelism and functions [page 2146]
Designer Guide: Distributed data flow execution [page 301]
Catch error functions [page 847]

3.6.2.1 Database and application functions

You can import the metadata for database and application functions and use them in Data Services applications. You can also import stored functions and procedures.

At run time, Data Services passes the appropriate information to the database or application from which the function was imported.

The metadata for a function includes the input and output parameters and their data types. If there are restrictions on data passed to the function, such as requiring uppercase values or limiting data to a specific range, you must enforce these restrictions in the input. You can either test the data before extraction or include logic in the data flow that calls the function.

Related Information

About procedures [page 1697]
Designer Guide: Imported stored function and procedure information [page 221]

3.6.3 Descriptions of built-in functions

This section describes each built-in function available in Data Services.

The following table lists the names and descriptions of functions, as well as the function's category in the function wizard and smart editor.

Function	Category	Description
abs	Math	Returns the absolute value of an input number.
add_months	Date	Adds a given number of months to a date.
ascii	String	Returns the decimal value of the first character for the given string using ASCII character set. If the character passed is not a valid ASCII character, -1 is returned.
avg	Aggregate	Calculates the average of a given set of values.
base64_decode	Miscellaneous	Returns the source data after decoding the base64-encoded input.
base64_encode	Miscellaneous	Returns the base64-encoded data in the engine locale character set.
before_image	Miscellaneous	Returns the before image value of a row. This function is available for the Map_Operation transform only and is applicable to UPDATE rows.
cast	Conversion	Returns a value in the cast data type.
ceil	Math	Returns the smallest integer value greater than or equal to an input number.
chr	String	Get character representation of provided ASCII value.
concat_date_time	Date	Returns a datetime from separate date and time inputs.
count	Aggregate	Counts the number of values in a table column.
count_distinct	Aggregate	Count the number of distinct non-null values in a table column.
current_configuration	Miscellaneous	Returns the name of the datastore configuration in use at runtime.
current_system_configuration	Miscellaneous	Returns the name of the system configuration in use at runtime. If no system configuration is defined, returns a NULL value.
dataflow_name	Miscellaneous	Returns the data flow name in which this call exists. If the call is not in a data flow, returns NULL.
datastore_field_value	Miscellaneous	Retrieves the value of a specified datastore field.
date_diff	Date	Returns the difference between two dates or times.
date_part	Date	Extracts a component of a given date.
day_in_month	Date	Determines the day in the month on which the given date falls.
day_in_week	Date	Determines the day in the week on which the given date falls.

Function	Category	Description
day_in_year	Date	Determines the day in the year on which the given date falls.
db_type	Miscellaneous	Returns the database type of the datastore configuration in use at runtime.
db_version	Miscellaneous	Returns the database version of the datastore configuration in use at runtime.
db_database_name	Miscellaneous	Returns the database name of the datastore configuration in use at runtime.
db_owner	Miscellaneous	Returns the real owner name for the datastore configuration that is in use at runtime.
decode	Miscellaneous	Returns an expression based on the first condition in the specified list that evaluates to TRUE.
decrypt_aes	Cryptographic	Decrypts the input string using the user-specified passphrase and key length using the AES algorithm.
decrypt_aes_ext	Cryptographic	Decrypts cipher text using the AES key generated using the specified passphrase and salt. The passphrase and salt must be the same as those used to encrypt the data.
double_metaphone	String	Encodes the input string using the Double Metaphone algorithm and returns a string.
encrypt_aes	Cryptographic	Encrypts the input string using the user-specified passphrase and key length using the AES algorithm.
encrypt_aes_ext	Cryptographic	Encrypts plain text and encodes it in base64 using the AES key generated by using the specified passphrase, salt, and key length. Given the same input, the encrypted output should be the same. The caller of this function must ensure that the space to hold encrypted text is at least 1.33 times larger than the original plain text.
error_timestamp	Miscellaneous (Can only be found when creating a script)	Returns the timestamp of the caught exception.
error_context	Miscellaneous (Can only be found when creating a script)	Returns the context of the caught exception. For example, "|Session datapreview_job|data flow debug_DataFlow|Transform Debug"
error_message	Miscellaneous (Can only be found when creating a script)	Returns the error message of the caught exception.
error_number	Miscellaneous (Can only be found when creating a script)	Returns the error number of the caught exception.

Function	Category	Description
exec	System	Sends a command to the operating system for execution.
extract_from_xml	Conversion	Extracts XML directly from single column in a database table, and converts it into its internal nested relational data model (NRDM). To access this function, you must open the function wizard from within a new function call.
file_exists	Miscellaneous	Checks to see if a given file or directory exists.
fiscal_day	Date	Converts a given date into an integer value representing a day in a fiscal year.
floor	Math	Returns the largest integr value less than or equal to an input number.
gen_row_num_by_group	Miscellaneous	Returns group row number of the record.
gen_row_num	Miscellaneous	Returns an integer value beginning with 1 then incremented sequentially by 1 for each additional call. This function can be used to generate a column of row IDs.
gen_uuid	Miscellaneous	Returns a unique varchar string identifier.
get_domain_description	Miscellaneous	Returns the description of a value when given the domain name and the value.
get_env	Environment	Returns a value for the specified environmental variable.
get_error_filename	Environment	Returns the full path and file name for the error log.
get_file_attribute	Miscellaneous	Returns date created, date modified, or size (in bytes) of a physical file.
get_monitor_filename	Environment	Returns the full path and file name for the monitor log.
get_trace_filename	Environment	Returns the full path and file name for the trace log.
greatest	Miscellaneous	Returns greatest of the list of one or more expressions.
host_name	Miscellaneous	Returns the name of the computer on which the job is executing.
ifthenelse	Miscellaneous	Allows conditional logic in mapping and selection operations.
index	String	Returns the index of a given word in a string.
init_cap	String	Changes the characters in a string to title case.
interval_to_char	Conversion	Returns a string representation of the interval.
is_group_changed	Miscellaneous	Returns 1 if the group is changed, 0 otherwise.
is_set_env	Environment	Verifies if the specified environment variable is set.

Function	Category	Description
is_valid_date	Validation	Indicates if an expression can be converted into a valid date value.
is_valid_datetime	Validation	Indicates if an expression can be converted into a valid datetime value.
is_valid_decimal	Validation	Indicates if an expression can be converted into a valid decimal value.
is_valid_double	Validation	Indicates if an expression can be converted into a valid double value.
is_valid_int	Validation	Indicates if an expression can be converted into a valid integer value.
is_valid_real	Validation	Indicates if an expression can be converted into a valid real value.
is_valid_time	Validation	Indicates if an expression can be converted into a valid time value.
isempty	Miscellaneous	Indicates if a nested table contains data.
isweekend	Date	Indicates that a date corresponds to Saturday or Sunday.
job_name	Miscellaneous	Returns the name of the job in which the call to this function exists.
job_run_id	Miscellaneous	Retrieves the job run ID for the current job execution.
julian	Date	Converts a date to its integer Julian value, the number of days between the start of the Julian calendar and the date.
julian_to_date	Conversion	Converts a Julian value to a date.
key_generation	Database	Generates keys for the specified table, after determining the appropriate starting value.
last_date	Date	Returns the last date of the month for a given date.
least	Miscellaneous	Returns the least in a list of one or more expressions.
length	String	Returns the number of characters in a given string.
literal	String	Returns an input constant expression without interpolation. Allows you to assign a pattern to a variable without interpolation.
ln	Math	Returns the natural logarithm of the given numeric expression.
load_to_xml	Conversion	Generates XML text from NRDM and loads it into a single database column (Assumes the database supports XML text in its columns).

Function	Category	Description
log	Math	Returns the base-10 logarithm of the given numeric expression.
long_to_varchar	Conversion	Converts a data type from long to varchar.
lookup	Lookup	Finds a value in one table or file based on values in a second table or file.
lookup_ext	Lookup	Finds data from a database table, flat file, or memory datastore table.
lookup_seq	Lookup	Finds a value in one table based on values in a second table or file, and ensures that the sequence matches.
lower	String	Changes the characters in a string to lowercase.
lpad	String	Pads a string with characters from a specified pattern.
lpad_ext	String	Pads a string with logical characters from a specified pattern.
ltrim	String	Removes specified characters from the start of a string.
ltrim_blanks	String	Removes blank characters from the start of a string.
ltrim_blanks_ext	String	Removes blank and control characters from the start of a string.
mail_to	System	Sends the specified e-mail message.
match_pattern	String	Matches whole input strings to simple patterns supported by Data Services. This function does not match substrings.
match_regex	String	Matches whole input strings to the pattern that you specify with regular expressions (regular expressions based on the POSIX standard) and flags. This function does not match substrings.
max	Aggregate	Returns the maximum value from a list.
min	Aggregate	Returns the minimum value from a list.
mod	Math	Returns the remainder when one number is divided by another.
month	Date	Determines the month in which the given date falls.
num_to_interval	Conversion	Converts a numeric value to an interval.
nvl	Miscellaneous	Replaces NULL values.
power	Math	Returns the value of the give expression to the specified power.
previous_row_value	Miscellaneous	Returns the column value of previous row.
print	String	Prints the given string to the trace log.

Function	Category	Description
pushdown_sql	Miscellaneous	Allows you to create dynamic WHERE clauses.
quarter	Date	Determines the quarter in which the given date falls.
raise_exception	Miscellaneous	Calling this function causes an exception to be generated.
raise_exception_ext	Miscellaneous	Same as raise_exception, but takes a second parameter for an exit code.
rand	Math	Returns a random number between 0 and 1.
rand_ext	Math	Returns a random number between 0 and 1.
replace_substr	String	Returns a string where every occurrence of a given search string in the input is substituted by the given replacement string.
replace_substr_ext	String	Takes an input string, replaces specified occurrences of a specified sub-string with a specified replacement and returns the result. You can also use this function to search for hexadecimal or reference characters.
repository_name	Miscellaneous	Returns a database connection string and owner name. For example: `beq-local.DBUser.` This is the ID for the repository from which the job is run.
round	Math	Rounds a given number to the specified precision.
rpad	String	Pads a string with characters from a given pattern.
rpad_ext	String	Pads a string with logical characters from a given pattern.
rtrim	String	Removes given characters from the end of a string.
rtrim_blanks	String	Removes blank characters from the end of a string.
rtrim_blanks_ext	String	Removes blank and control characters from the end of a string.
sap_openhub_processchain_execute	SAP	Starts the process chain that extracts data from an SAP NetWeaver Business Warehouse (BW) and loads the extracted data into an Open Hub Destination table.
sap_openhub_set_read_status	SAP	Sends the read status for the Open Hub table to SAP NetWeaver BW.
search_replace	String	Searches input parameters and replaces by matching criteria and values specified by search table.
set_cdc_checkpoint	Miscellaneous	Sets a check-point for a Microsoft SQL Server changed-data-capture (CDC method) job for data flows that run in a WHILE loop.
set_env	Environment	Sets an environmental variable temporarily to a specified value.

Function	Category	Description
sleep	Miscellaneous	Suspends the execution of the data flow or work flow from where it is called.
soundex	String	Encodes the input string using the Soundex algorithm and returns a string. Use when you want to push down the function to the database-level.
sql	Database	Runs a SQL operation in the specified database.
sqrt	Math	Returns the square root of the given expression.
smtp_to	System	Sends the specified e-mail message using the SMTP protocol.
substr	String	Returns a specific portion of a string starting at a given point in the string.
sum	Aggregate	Calculates the sum of a given set of values.
sysdate	Date	Returns the current date as listed by the Job Server's operating system.
system_user_name	Miscellaneous	Returns the user name used to log into the Job Server's operating system.
systime	Time	Returns the current time as listed by the operating system.
table_attribute	Miscellaneous	Retrieves the value of a specified table attribute.
to_char	Conversion	Converts a date or numeric type to a string.
to_date	Conversion	Converts a string to a date.
to_decimal	Conversion	Converts a varchar to a decimal.
to_decimal_ext	Conversion	Converts a varchar to a decimal, including precision as a parameter.
total_rows	Database	Returns the number of rows in a particular table in a datastore.
trunc	Math	Truncates a given number to the specified precision.
truncate_table	Miscellaneous	Allows you to explicitly expunge data from a memory table.
upper	String	Changes the characters in a string to uppercase.
varchar_to_long	Conversion	Converts a data type from varchar to long.

Function	Category	Description
wait_for_file	Miscellaneous	Returns the existing files that match the input file pattern.
week_in_month	Date	Determines the week in the month in which the given date falls.
week_in_year	Date	Determines the week in the year in which the given date falls.
WL_GetKeyValue	String	Returns the value of a given keyword in Web log search strings.
word	String	Returns one word out of a string.
word_ext	String	Returns the word identified by its position in a delimited string.
workflow_name	Miscellaneous	Returns the work flow in which this call exists. Returns the name of the inner most work flow in cases where several work flows enclose this function call. If no work flow is found, returns job name.
year	Date	Determines the year in which the given date falls.

Related Information

Catch error functions [page 847]

3.6.3.1 abs

Returns the absolute value of a number.

Syntax

```
abs(<num>)
```

Return value

decimal, double, int, or real

The absolute value of the given number, **<num>** . The type of the return value is the same as the type of the original number.

Where

<num>	The source number.

Example

Function	Results
`abs(12.12345)`	`12.12345`
`abs(-12.12345)`	`12.12345`

3.6.3.2 add_months

Adds a given number of months to a date.

Syntax

```
add_months(<original_date, months_to_add>)
```

Return Value

date

Where

<original_date>	Specify the starting year.month.date.
<months_to_add>	Number of months to add to the original date.

Details

The **<months_to_add>** can be any integer. If **<original_date>** is the last day of the month or if the resulting month has fewer days than the day component of **<original_date>**, then the result is the last day of the resulting month. Otherwise, the result has the same day component as **<original_date>**.

Function	Results
`add_month('1990.12.17', 1)`	`'1991.01.17'`
`add_month('2001.10.31',4)`	`'2002.2.28'`

3.6.3.3 ascii

Returns decimal value of ASCII code of the first character in the input string.

Syntax

```
ascii(input_string)
```

Return Value

Int

Where

<input_string>	The source string.

Details

Returns the decimal value of the ASCII code of the first character in the input string. Returns -1 if the first character is not a valid ASCII character.

Example

Function	Results
`ascii('AaC')`	65

3.6.3.4 avg

Calculates the average of a given set of values.

Syntax

```
avg(<value_list>)
```

Return value

decimal, double, int, or real

The calculated average of **`<value_list>`**. The average is calculated to the same precision as the input value.

Where

`<value_list>`	The source values for which to calculate an average, such as a table column.

Example

To calculate the average of values in the salary column of a table, use the avg function in a query:

- In the *Mapping* tab of the query editor, enter:

  ```
  avg(SALARY)
  ```

- In the *Group By* tab in the query editor, specify the columns for which you want to group the salary, such as the department column. For each unique set of values in the group by list, such as each unique department, Data Services calculates the average salary.

3.6.3.5 base64_decode

Returns the source data after decoding the base64-encoded input.

Syntax

```
base64_decode(<base64-encoded input>, 'UTF-8')
```

Return Value

varchar or blob

Returns the source data after decoding the base64-encoded input. If the input is NULL or the size of the data is 0, Data Services returns NULL. Otherwise, it returns the base64-decoded data that conforms to RFC 2045.

Where

<base64-encoded input >	The base64-encoded input data. Supports varchar and blob data types.
UTF-8	The code page of the output data. UTF-8 is required for Data Integrator version 11.7.3. This paramater is not required when the input data type is blob.

Related Information

base64_encode [page 1527]

3.6.3.6 base64_encode

Returns the base64-encoded data in the engine locale character set.

Syntax

```
base64_encode(<input data>, 'UTF-8')
```

Return Value

varchar or blob

Returns base64-encoded data. If the input data is NULL or the size is 0, Data Services returns NULL. Otherwise, it returns the base64-encoded data that conforms to RFC 2045.

Where

`<input data>`	The input data that needs to be encoded to base64. Supports varchar and blob data types.
UTF-8	The code page of the input data. UTF-8 is required for Data Integrator version 11.7.3. This parameter is not required when the input data type is blob.

Related Information

base64_decode [page 1527]

3.6.3.7 before_image

Retrieves the before image value of a row. This function is available for the Map_Operation transform only and is applicable to UPDATE rows.

Note

You cannot use quoted strings in the parameter value for before_image. Because of this limitation, you cannot use global variable. For example, `before_image(T1.Col1)` is valid, but `before_image('T1.Col1')` is not.

Syntax

```
before_image(<column_name>)
```

Return value

Inline

The return value is the same as the input column value. For example, if the input column (column_name) is an integer, the software returns an integer.

Where

<column_name>	The name of the table column.

Example

```
before_image(Epml.SALARY)
```

Related Information

Map_Operation [page 1436]

3.6.3.8 cast

Explicitly converts an expression of one data type to another.

Syntax

```
cast(expression,data_type)
```

Return Value

Returns the same value in data_type.

Where

<expression>	Input expression that needs to be cast to target data type.
<data_type>	Target data type. This must be a built-in Data Services data type and specified as a constant string, for example, 'decimal`(28,7)`'. See the Target Data Type table below for syntax.

Details

The cast function explicitly converts the value of the first parameter into the built-in data type specified in the second parameter. The Cast Type Compatibility Matrix shows all explicit data type conversions that are valid for this function.

Cast type compatibility matrix

From / To	Date	Date time	Decimal	Double	Int	Interval	Real	Time	Time stamp	Varchar
Date	X	X							X	X
Date time	X	X						X	X	X
Decimal			X	X	X	X	X			X
Double			X	X	X	X	X			X
Int			X	X	X	X	X			X
Interval			X	X	X	X	X			X
Real			X	X	X	X	X			X
Time		X						X	X	X
Time stamp	X	X						X	X	X
Varchar	X	X	X	X	X	X	X	X	X	X

Target data type syntax

Data type	Syntax
varchar	'varchar(length)'
decimal	'decimal(precision,scale)'
integer	'int'
real	'real'
double	'double'
timestamp	'timestamp'
datetime	'datetime'
date	'date'

Target data type syntax	
Data type	**Syntax**
time	'time'
interval	'interval'

Table 174: Example

Input	Output
cast('20.3','decimal(3,1)')	20.3

3.6.3.9 chr

Converts a decimal ASCII code to a character.

Syntax

```
chr (integer_expression)
```

Return Value

ASCII character

Where

integer_expression	Integer from 0 through 255. Returns NULL if the integer expression is not in this range.

Details

This function returns the character associated with the specified ASCII code decimal number. If you specify a value of less than 0 or greater than 255 for the integer_expression parameter, SAP Data Services returns a NULL value. Use chr to insert control characters into character strings. For example, `chr(9)` can be used to insert <tab>.

Example

Function	Results
`chr(65)`	'A'

3.6.3.10 ceil

Returns the smallest integer value greater than or equal to a number.

Syntax

```
ceil(<num>)
```

Return value

decimal, double, int, or real

The indicated integer, cast as the same type as the original number, **<num>**.

Where

<num>	The source number.

Example

Function	Results
`ceil(12.12345)`	`13.00000`
`ceil(12)`	`12`
`ceil(-12.223)`	`-12.000`

3.6.3.11 concat_date_time

Returns a datetime from separate date and time inputs.

Syntax

```
concat_date_time(<date, time>)
```

Return value

datetime

The datetime value obtained by combining the inputs.

Example

```
concat_date_time(MS40."date",MS40."time")
```

3.6.3.12 count

Counts the number of values in a group.

Syntax

```
count(<column>)
```

Return value

int

The number of rows in the column that have a non-NULL value.

Where

`<column>`	The column in the input table in which to count values.

Example

To determine the number of customers located in a specific sales region, use the count function with a filter defined in the *Where* tab of the query editor. The following WHERE clause selects the rows in the REGION column with the value `TX`:

```
REGION = "TX"
```

With the target column selected, enter the count function in the *Mapping* tab of the editor:

```
count(REGION)
```

3.6.3.13 count_distinct

Returns the number of distinct non-null values in a group.

Syntax

```
count_distinct (expression)
```

Return Value

Integer

Where

expression	Any valid expression of any type except NRDM or long data type.

Example

In a customer table, the customer's region is stored in a column named `REGION`. To count the number of distinct regions the customers come from, use the count_distinct function with a filter defined in the *Where* tab of the query editor. Enter the count_distinct function in the *Mapping* tab of the editor, as follows:

```
count_distinct(REGION)
```

Input

Name	Region	Country
Cust 1	East	US
Cust 2	East	US
Cust 3	West	US
Cust 4	East	France

Output

```
count_distinct(REGION) = 2
```

If you need to calculate the number of distinct regions per country, add the country column to the group by clause, as follows:

`count_distinct(REGION)`	Country
2	US
1	France

Note

If you want to provide more resources to execute the `count_distinct` function, select *Run as a separate process*. This option creates a separate sub data flow process for the `count_distinct` function when Data Services executes the data flow.

Related Information

Designer Guide: Distributed data flow execution [page 301]

3.6.3.14 current_configuration

Returns the name of the datastore configuration that is in use at runtime. If the datastore does not support multiple configurations, for example, the datastore is a memory datastore, the name of the datastore is returned instead.

Syntax

```
current_configuration(ds_name)
```

Return Value

varchar

Where

<ds_name>	The name you enter when you create the datastore.

Example

Create a job and add a script with, for example, the following line.

```
print('Datastore Configuration used at runtime: [current_configuration()]')
```

Returns, for example, the following to the trace log:

```
Datastore configuration used at runtime: Test_DS
```

3.6.3.15 current_system_configuration

Returns the name of the system configuration used at runtime. If no system configuration is defined, returns a NULL value.

Syntax

```
current_system_configuration()
```

Return Value

varchar

Example

Create a job and add a script with, for example, the following line.

```
print('System Configuration used at runtime: [current_system_configuration()]')
```

This line returns, for example, the following to the trace log:

```
System configuration used at runtime: Production
```

3.6.3.16 dataflow_name

Returns the data flow name in which this call exists. If the call is not in a data flow, returns NULL.

Syntax

```
dataflow_name()
```

Return Value

varchar

Example

```
print('Data Flow Name: [dataflow_name()]')
```

3.6.3.17 datastore_field_value

Retrieves the value of a specified datastore field.

Syntax

```
datastore_field_value(<ds_name, field_name>)
```

Return Value

varchar

Where

`<ds_name>`	The name you enter when you create the datastore.
`<field_name>`	The name of the field.

Details

The `<field_name>` should match the name seen in the language of the datastore. In the datastore editor click the Show ATL button to see the valid field names. If a specified field is not found or the datastore is invalid, NULL is returned. If the `<field_name>` is 'password' NULL is returned.

Example

Function	Results
`datastore_field_value('mssql', 'sql_server_database')`	`'DBUser'`

3.6.3.18 date_diff

Returns the difference between two dates or times.

Syntax

```
date_diff(<date1, date2, fmt_str>)
```

Return Value

int

Where

`<date1, date2 >`		
`<fmt_str>`	The string describing the format of the dates. Choose from the following values:	
	D	Day
	H	Hours
	M	Minutes
	S	Seconds
	MM	Months

	YY	Years

Details

This function is equivalent to interval_to_char(date1 - date2, 'fmt_str'), except if date1 is smaller than date2, the date_diff function returns a positive value.

Example

Function	Results
`date_diff(start_date,sysdate(),'D')`	The number of days between the date in the column start_date and today's date.
`date_diff(start_time,systime(),'M')`	The number of minutes between the time in the column start_time and the current time.

3.6.3.19 date_part

Extracts a component of a given date.

Syntax

```
date_part(<in_date, fmt_str>)
```

Return Value

int

Where

<in_date>	The input date.	
<fmt_str>	The string describing the format of the extracted part of the date. Choose from the following values:	
	YY	Year

	MM	Month
	DD	Day
	HH	Hours
	MI	Minutes
	SS	Seconds

Details

This function takes in a datetime and extracts the component requested as an integer.

Note

`<Year>` is displayed as four digits, not two.

Example

Function	Results
`date_part('1990.12.31', 'YY')`	`1990`
`date_part('1991.01.17 23:44:30', 'SS')`	`30`

3.6.3.20 day_in_month

Determines the day in the month on which the input date falls.

Syntax

```
day_in_month(<date1>)
```

Return value

int The number from 1 to 31 that represents the day in the month that `<date1>` occurs.

Where

`<date1>`	The source date.

This function extracts the day component from the date value.

Example

Function	Results
`day_in_month(to_date('Jan 22, 1997','mon dd, yyyy'))`	22
`day_in_month(to_date('02/29/1996','mm/dd/yyyy'))`	29
`day_in_month(to_date('1996.12.31','yyyy.mm.dd'))`	31

3.6.3.21 day_in_week

Determines the day in the week on which the input date falls.

Syntax

```
day_in_week(<date1>)
```

Return value

int

The number from 1 (Monday) to 7 (Sunday) that represents the day in the week that `<date1>` occurs.

Where

`<date1>`	The source date.

This function allows you to categorize dates according to the day of the week the date falls upon. For example, all dates for which this function returns a "3" occur on Wednesday.

Example

Function	Results
`day_in_week(to_date('Jan 22, 1997','mon dd, yyyy'))`	3 (Wednesday)
`day_in_week(to_date('02/29/1996','mm/dd/yyyy'))`	4 (Thursday)
`day_in_week(to_date('1996.12.31','yyyy.mm.dd'))`	2 (Tuesday)

3.6.3.22 day_in_year

Determines the day in the year on which the input date falls.

Syntax

```
day_in_year(<date1>)
```

Return value

int

The number from 1 to 366 that represents the day in the year that **<date1>** occurs.

Where

<date1>	The source date.

Example

Function	Results
`day_in_year(to_date('Jan 22, 1997','mon dd, yyyy'))`	22
`day_in_year(to_date('02/29/1996','mm/dd/yyyy'))`	60

Function	Results
`day_in_year(to_date('1996.12.31','yyyy.mm.dd'))`	`366` (1996 was a leap year.)

3.6.3.23 db_type

Returns the database type of the datastore configuration in use at runtime.

This function is useful if your datastore has multiple configurations. For example, you can use this function in a SQL statement instead of using a constant. This allows the SQL statement to use the correct database type for each job run no matter which datastore configuration is in use.

Syntax

```
db_type(<ds_name>)
```

Return Value

varchar

Possible db_type() return values for datastore types are as follows:

Datastore Types	Possible db_type() Return Value
Adapter	Adapter
Database	Attunity_Connector, DB2, Informix, Memory, Microsoft_SQL_Server, ODBC, Oracle, SAP, SAP_BW, SQL_Anywhere, SAP Sybase (for SAP Sybase ASE), Sybase_IQ, Teradata
JDE One World	DB2, Microsoft_SQL_Server, ODBC, or Oracle
JDE World	ODBC
Oracle Applications	Oracle
PeopleSoft	Microsoft_SQL_Server, or Oracle
SAP Applications	SAP
SAP BW Source	SAP
SAP BW Target	SAP_BW
SAP Master Data Services	HANA

Datastore Types	Possible db_type() Return Value
Siebel	DB2, Microsoft_SQL_Server, or Oracle

Where

<ds_name>	The datastore name you enter when you create the datastore.

Example

If you have a SQL transform that performs a function that has to be written differently for database types, you can tell SAP Data Services what to do if the database type is Oracle.

In this example, the sql() function is used within a script.

```
IF (db_type('sales_ds') = 'Oracle')
BEGIN
    IF (db_version('sales_ds') = 'Oracle 9i')
        $sql_text = '…';
    ELSE
           $sql_text = '…';
END
Sql('sales_ds', '{$sql_text}');
```

3.6.3.24 db_version

Returns the database version of the datastore configuration in use at runtime. This function is useful if your datastore has multiple configurations. For example, you can use this function in a SQL statement instead of using a constant. This allows the SQL statement to use the correct database version for each job run no matter which datastore configuration is in use.

Syntax

```
db_version(<ds_name>)
```

Return Value

varchar

Possible db_version() return values are:

Database type	Version
Oracle	Currently supported versions
Microsoft SQL Server	Currently supported versions
DB2 UDB	Currently supported versions
Informix IDS	Currently supported versions
SQL Anywhere	Currently supported versions
SAP Sybase ASE	Currently supported versions
SAP Sybase IQ	Currently supported versions
Teradata	Currently supported versions
""	An empty string is returned for any other database type

Where

<ds_name>	The datastore name you enter when you create the datastore.

Example

If you have a SQL transform that performs a function that has to be written differently for different versions of Oracle, you can tell Data Services which text to use for each database version. In this example, the sql() function is used within a script.

```
IF (db_type('sales_ds') = 'Oracle')
BEGIN
    IF (db_version('sales_ds') = 'Oracle 9i')
        $sql_text = '…';
    ELSE
           $sql_text = '…';
END
Sql('sales_ds', '{$sql_text}');
```

3.6.3.25 db_database_name

Returns the database name of the datastore configuration in use at runtime.

This function is useful if your datastore has multiple configurations and is accessing an MS SQL Server or SAP Sybase ASE database. For a datastore configuration that is using either of these database types, you enter a database name, when you create a datastore. This function returns that database name.

For example, master is a database name that exists in every Microsoft SQL Server and SAP Sybase ASE database. However, if you use different database names, you can use this function in, for example, a SQL statement instead of using a constant. This allows the SQL statement to use the correct database name for each job run no matter which datastore configuration is in use.

This function returns an empty string for datastore configurations without MS SQL Server or SAP Sybase ASE as the Database Type.

Syntax

```
db_database_name(<ds_name>)
```

Return Value

varchar

Where

<ds_name>	The datastore name you enter when you create the datastore.

Example

If you have a SQL transform that performs a function that has to be written differently for different versions of database types, you can tell Data Services which text to use for each database version. In this example, the sql() function is used within a script.

```
IF (db_type('sales_ds') = 'DB2')
    $sql_text = '…';
ELSE
BEGIN
    IF (db_type('sales_ds') = 'MicroSoft_SQL_Server')
        $db_name = db_database_name('sales_ds');
        $sql_text = '…';
END
Sql('sales_ds', '{$sql_text}');
```

3.6.3.26 db_owner

Returns the real owner name for the datastore configuration that is in use at runtime.

This function is useful if your datastore has multiple configurations because with multiple configurations, you can use alias owner names instead of database owner names. By using aliases instead of real owner names, you limit the amount of time it takes to port jobs to different environments.

For example, you can use this function in a SQL statement instead of using a constant. This allows the SQL statement to use the correct database owner for each job run no matter which datastore configuration is in use.

Syntax

```
db_owner(<ds_name>, <alias_name>)
```

Return Value

varchar

Where

`ds_name`	The datastore name that you entered when you created the datastore.
`alias_name`	The name of the alias that you created in the datastore, then mapped to the real owner name when you created a datastore configuration.

Example

```
$real_owner = db_owner('sales_ds', 'sales_person');
```

3.6.3.27 decode

Returns an expression based on the first condition in the specified list of conditions and expressions that evaluates to TRUE.

Syntax

```
decode(<condition_and_expression_list,><default_expression>)
```

Return value

<expression> or **<default_expression>**

Returns the value associated with the first **<condition>** that evaluates to TRUE. The data type of the return value is the data type of the first **<expression>** in the **<condition_and_expression_list><.>**

If the data type of any subsequent **<expression>** or the **<default_expression>** is not convertible to the data type of the first **<expression>**, Data Services produces an error at validation. If the data types are convertible but do not match, a warning appears at validation.

Where

<condition_and_expression_list>	A comma-separated list of one or more pairs that specify a variable number of conditions. Each pair contains one **<condition>** and one **<expression>** separated by a comma. You must specify at least one **<condition>** and **<expression>** pair. The **<condition>** evaluates to TRUE or FALSE. The **<expression>** is the value that the function returns if the **<condition>** evaluates to TRUE.
<default_expression>	An expression that the function returns if none of the conditions in **<condition_and_expression_list>** evaluate to TRUE. You must specify a **<default_expression>**.

Details

The `decode` function provides an easier way to write nested `ifthenelse` functions. In nested `ifthenelse` functions, you must write nested conditions and ensure that the parentheses are in the correct places, as the following example shows:

```
ifthenelse ((EMPNO = 1), '111',
  ifthenelse((EMPNO = 2), '222',
      ifthenelse((EMPNO = 3), '333',
        ifthenelse((EMPNO = 4), '444',
          'NO_ID'))))
```

In the `decode` function, you list the conditions, as the following example shows. Therefore, `decode` is less error prone than nested `ifthenelse` functions.

```
decode ((EMPNO = 1), '111',
(EMPNO = 2), '222',
(EMPNO = 3), '333',
(EMPNO = 4), '444',
'NO_ID')
```

To improve performance, Data Services pushes this function to the database server when possible. Thus, the database server, rather than Data Services, evaluates the `decode` function.

Use this function to apply multiple conditions when you map columns or select columns in a query. For more flexible control over conditions in a script, use the IF keyword in the Data Services scripting language.

If a condition compares a varchar value with trailing blanks, the decode function ignores the trailing blanks.

To compare a NULL value (NULL constant or variable that contains a NULL constant), use the IS NULL or IS NOT NULL operator. If you use the Equal (=) or Not equal to (<>) operator, the comparison against a NULL value always evaluates to FALSE.

Example

Function	Results

`decode((COUNTRY = 'FRANCE'),`
`'French',`
`  (COUNTRY = 'GERMANY'), 'German',`
`  (COUNTRY = 'ITALY'), 'Italian',`
`  (COUNTRY = 'TAIWAN'), 'China',`
`  (COUNTRY = 'USA'), 'America',`
`  (COUNTRY IS NULL), 'Unknown',`
`      'Others')`
	If the value in the column COUNTRY is `FRANCE`, the value returned is `French`. If COUNTRY is `NULL`, the value returned is `Unknown`. If COUNTRY does not contain any of the values listed, the `decode` function returns the value `Others`.

3.6.3.28 decrypt_aes

This function decrypts the input string using the user-specified passphrase and key length using the AES algorithm.

Syntax

```
decrypt_aes(<encrypted_input_string, passphrase, key_length_in_bits>)
```

Return value

Returns plain string as varchar.

In case of a failure, the function throws an exception of type execution error which results in termination of the job. You can catch the exception by using try/catch handlers.

If the encrypted input string is empty, then the return value is an empty string.

If the encrypted input string is NULL, then the return value is NULL.

Where

<encryptedinput_string>	A varchar input string to be decrypted.
<passphrase>	A varchar character string.
<key_length_in_bits>	An int value of 128, 192, or 256.

Example

For security purposes, you should secure the passphrase in a database and read it using a sql() function into a local or global variable. Then you can pass the variable to the passphrase parameter.

```
 #read the passphrase
from a secured source such as a database
$G_passphrase = sql('PASSWORD_DATASTORE', 'select PASSPHRASE from PASSWORD');
encrypt_aes(SOURCE.SSN,
$G_passphrase, 128);
```

Similar to other string functions, this function can be called from a custom function, in the column mapping of a Query transform or in a script in the work flow.

3.6.3.29 decrypt_aes_ext

This function decrypts the input string using the user-specified passphrase, salt, and key length using the AES algorithm. The passphrase and salt must be the same as those used to encrypt the data.

It generates an AES key of the specified key length using the specified passphrase and the key generation algorithm PKCS5_PBKDF2_SHA256. This key is used for decrypting the encrypted input string.

Syntax

```
decrypt_aes_ext(<Varchar Encrypted_input_string, Varchar Passphrase, Varchar
salt, Int Key_length_in_bits>)
```

Return value

Returns plain string as varchar.

In case of a failure, the function throws an exception of type execution error, which results in the termination of the job. You can catch the exception by using try/catch handlers.

If the encrypted input string is empty, then the return value is an empty string.

If the encrypted input string is NULL, then the return value is NULL.

If you fail to provide the same passphrase and key length used for encryption to this function, then the call does not fail but instead returns an incorrect output.

Where

<Encrypted_input_string>	A varchar input string to be decrypted.
<Passphrase>	A varchar character string with at least one character.
<Salt>	A varchar that must be exactly eight ASCII characters.
<Key_length_in_bits>	An int value of 128, 192, or 256.

Example

For security purposes, you should secure the passphrase and salts in a database and read it using a sql() function into a local or global variable. Then you can pass the variable to the passphrase parameter.

```
#read the passphrase from a secured source such as a database
$G_passphrase = sql('PASSWORD_DATASTORE', 'select PASSPHRASE from PASSWORD');
$G_salt = sql('PASSWORD_DATASTORE', 'select SALT from PASSWORD');

decrypt_aes_ext(ENCRYPTED.SSN, $G_passphrase, $G_salt, 128);
```

Similar to other string functions, this function can be called from a custom function, in the column mapping of a Query transform, or in a script in the work flow.

3.6.3.30 double_metaphone

Encodes the input string using the Double Metaphone algorithm and returns a string.

Syntax

```
double_metaphone(<input_str, alternate, return_if_empty>)
```

Return Value

varchar

Returns the string containing the double metaphone encoding of the input string. The length of the return string depends on the length of the input string, but it is always shorter than the input string.

Where

<input_str>	The source string to encode.

`<alternate>`	A flag to control how encoded strings are returned. When input as 0, return the primary encoding. If there is no primary encoding, then return null or the input string, depending on how `<return_if_empty>` is set. When the input is not 0, return the alternate encoding. If there is no alternate encoding, then return null or the input string, depending on how `<return_if_empty>` is set. When the parameter is null or invalid (non-numeric), it is defaulted to 0.
`<return_if_empty>`	A flag to determine whether to return null or the input string when there is no encoding. When input as 0, return null. Otherwise, return the input string when there is no encoding. When the parameter is null or invalid (non-numeric), it is defaulted to 1. When input is empty, there is no primary or secondary encodings. When `<return_if_empty>`=0, then return null. When `<return_if_empty>`=1, then return the empty string.

Details

Only use this function for input strings in English. Non-English characters are ignored.

When input is null, then return is null.

When the second or third parameter has an invalid value, default to 0 and 1, respectively.

Example

Function	Result
`Print(double_metaphone('Hello',0,0);`	Prints the double metaphone of the word “Hello.”
`double_metaphone($VAR, 1,1);`	If the string stored in $VAR does not have encoding available, then return the original string.
`double_metaphone($VAR,'a','b');`	Returns the primary double metaphone encoding or the variable $VAR when the primary encoding does not exist.

3.6.3.31 encrypt_aes

This function encrypts the input string using the user-specified passphrase and key length using the AES algorithm.

Syntax

```
encrypt_aes(<input_string, passphrase, key_length_in_bits>)
```

Return value

Returns encrypted string as varchar. The size of the encrypted string is usually twice as large as the size of plain text, therefore you must have enough space to hold the encrypted string.

In case of a failure, the function throws an exception of type execution error which results in termination of the job. You can catch the exception by using try/catch handlers.

If the input string is empty, then the return value is an encrypted string. The encrypted string is different for multiple calls of encrypt_aes() function with an empty input string.

If the input string is NULL, then the return value is NULL.

Where

<input_string>	A varchar input string to be encrypted.
<passphrase>	A varchar character string.
<key_length_in_bits>	An int value of 128, 192, or 256.

Example

For security purposes, you should secure the passphrase in a database and read it using a sql() function into a local or global variable. Then you can pass the variable to the passphrase parameter.

```
 #read the passphrase
from a secured source such as a database
$G_passphrase = sql('PASSWORD_DATASTORE', 'select PASSPHRASE from PASSWORD');
encrypt_aes(SOURCE.SSN,
$G_passphrase, 128);
```

Similar to other string functions, this function can be called from a custom function, in the column mapping of a Query transform, or in a script in the work flow.

3.6.3.32 encrypt_aes_ext

This function encrypts the input string using the user-specified passphrase, salt, and key length using the AES algorithm.

It generates an AES key of specified key length using the specified passphrase, salt, and the key generation algorithm PKCS5_PBKDF2_SHA256. This key is used for encrypting the input string.

Syntax

```
encrypt_aes_ext(<Varchar Input_string, Varchar Passphrase, Varchar salt, Int
Key_length_in_bits>)
```

Return value

Returns encrypted string as base64 encoded string. The size of the encrypted string is 1.3 times larger than the size of plain text. Therefore you must have enough space to hold the encrypted string.

In case of a failure, the function throws an exception of type execution error, which results in the termination of the job. You can catch the exception by using try/catch handlers.

If the input string is empty, then the return value is empty.

If the input string is NULL, then the return value is NULL.

Where

<Input_string>	A varchar input string to be encrypted.
<Passphrase>	A varchar character string.
<Salt>	A varchar that must be exactly eight ASCII characters.
<Key_length_in_bits>	An int value of 128, 192, or 256.

Example

For security purposes, you should secure the passphrase and salts in a database and read it using a sql() function into a local or global variable. Then you can pass the variable to the passphrase parameter.

```
#read the passphrase from a secured source such as a database
$G_passphrase = sql('PASSWORD_DATASTORE', 'select PASSPHRASE from PASSWORD');
$G_salt = sql('PASSWORD_DATASTORE', 'select SALT from PASSWORD');

encrypt_aes_ext(SOURCE.SSN, $G_passphrase, $G_salt, 128);
```

Similar to other string functions, this function can be called from a custom function, in the column mapping of a Query transform, or in a script in the work flow.

3.6.3.33 error_context

Returns the context of the caught exception.

Syntax

```
error_timestamp()
```

Return value

varchar 512

Example

"|Session datapreview_job|Dataflow debug_DataFlow|Transform Debug"

Related Information

Catch error functions [page 847]

3.6.3.34 error_message

Returns the error message of the caught exception.

Syntax

```
error_message()
```

Return value

varchar 512

Related Information

Catch error functions [page 847]

3.6.3.35 error_number

Returns the error number of the caught exception.

Syntax

```
error_number()
```

Return value

int

Related Information

Catch error functions [page 847]

3.6.3.36 error_timestamp

Returns the timestamp of the caught exception.

Syntax

```
error_timestamp()
```

Return value

timestamp

Related Information

Catch error functions [page 847]

3.6.3.37 exec

Sends a command to the operating system for execution. With this function, you can add a program to a Data Services job.

Syntax

```
exec(<command_file, parameter_list, flag>)
```

Return value

Varchar(1020)

Returns up to 1020 characters that depend on the value of **<flag>** .

Where

<command_file>	A string that indicates the location and file name to execute. This string is relative to the Job Server location. It can be an absolute or relative path. The files and directories in the path must be accessible from the Job Server's computer. The **<command_file>** can be a Windows batch file, a UNIX shell script, or a binary executable. To run other interpreted scripts, the **<command_file>** must be the name of the command interpreter (e.g., 'perl') and the script must be the first parameter in the **<parameter_list>** .
<parameter_list>	A string that lists the values to pass as arguments to the command file. Separate parameters with spaces. When passing no parameters to an executable, enter an empty string (' ').
<flag>	An integer that specifies what information appears in the return value string, and how to respond upon error—how to respond if **<command_file>** cannot be executed or exits with a nonzero operating system return code.

Table 175: Exec function flags

Flag	If successful, returns:	On error:	Notes:
0	Standard output	Raises an exception: `System function failure.`	
1	NULL string	Raises an exception: `System function failure.`	Use this flag to track error states in either of the following cases:

Flag	If successful, returns:	On error:	Notes:
			• The command never produces output • The calling job does not need output
2	Standard output	NULL string	Use this flag if you do not intend to track the status of the command other than the presence or absence of output.
3	NULL string	NULL string	
4	Standard output	Data Services error message string	Refer to "Details "
5	NULL string	Data Services error message string	Refer to "Details "
8	The concatenation of the return code and the combined stdout and stderr (standard error).	Returns the concatenation of the return code and the combined stdout and stderr (standard error).	Refer to "Details "
256	NULL string	NULL string	Use this flag if you want your program to run independently of Data Services. Unlike flags 0-8, if you use flag 256, Data Services will not wait until the command (executable program) completes before continuing with job processing. In this case, the command runs independently of Data Services and stdout, stderr, and return code cannot be returned. Raises an exception (`System function failure)` if the program cannot be launched (e.g., program file not found).

Details

- The program that this function executes must not wait for any user input (e.g., a prompt for a password). For flags 0-8, Data Services waits for the program to complete, therefore if the program hangs for input Data Services will hang also. For flag 256, Data Services will continue if the program hangs for input.
- For flags 4 and 5, the return value format for a Data Services error message string is:

  ```
  'error-number: error-message-text'
  ```

  where the first field is exactly 7 characters wide, and the second begins at index 10. If the program cannot be executed, the error number is 50307. If the program exits with a non-zero return code, the error number is 50306. The text is from Business Objects' errormessage.txt. For example:

  ```
  '  50306: Function <exec> failed to execute program 'foo.exe'. Program
  terminated with exit code 3.'
  ```
- For flag 8, the return value format is:

  ```
  'return-code: stdout-and-stderr'
  ```

  where the first field is exactly 7 characters wide and the second begins at index 10. The return code is produced by the program. Zero indicates success. Consult your program's documentation to determine the meaning of other codes.
  For example:

  - `' 0: 8 file(s) copied.'`
  - `' 1: The system cannot find the file specified.'`
  - `' 1: a.tmp -> /usr/tmp/a.tmp cp: *.lcl: The system cannot find the file specified.'`
  - `' -2: manmix(): fatal application error.'`

  The 7-character format enables you to easily extract the first field (the return code from the executed command) as a string of digits (which Data Services automatically converts to an integer wherever necessary), and the second field as a regular string. For example:

  - In a script:

    ```
    $foo = exec('foo.bat', ' ', 8);
    ```

    ```
    $foo_rc = substr($foo, 1, 7);
    ```

    ```
    $foo_txt = substr($foo, 10, 1020);
    ```
  - In a data flow, map

    ```
    exec('foo.bat', '', 8)
    ```

    to an output column “foo” in a query. Then in a subsequent query, refer to that column's components in a mapping or WHERE clause. For example:

    ```
    substr(query.foo, 1, 7);
    ```

    ```
    substr(query.foo, 10, 1020);
    ```

Use of remote shells

Use a remote shell to run a command elsewhere on the network:

- The **`<command_file>`** named in an exec call can be 'rsh' on either Windows or UNIX systems to invoke the remote shell facility. This is a means of running a command on a machine elsewhere on the network. For example:
  - `exec('rsh', '<RemoteMachineName> <CommandToRunRemotely> <CmdArg1> <CmdArg2>', 0);`
  - `exec('rsh', '<RemoteBox> -l<RemoteUser> <RemoteCommand> <CmdArg>', 3);`

  Invoke the remote shell facility sparingly, as the remote connection setup, remote authentication, and increased message traffic reduce performance.
- For **`<flag>`** values 4, 5, and 8, the return code which Data Services receives is that of the rsh (or remsh) command (i.e., 0 if it successfully gets a remote connection and authorization, nonzero otherwise). There is no relation between this and the return value of the remote command (this is inherent in the remote shell mechanism on all the operating systems). To work around this, wrap the remote command in a `.bat` file (Windows) or shell script (UNIX) which will get the command's return code (%errorlevel% if Windows, $? if UNIX), and print it to stdout or stderr. For example:
  - `exec('rsh', '<RemoteMachineName> <remcmdWrapper>.bat <CmdArg1> <CmdArg2>', 8);`
  - `exec('rsh', '<RemoteBox> -l<RemoteUser> /usr/acta/<remcmdWrapper> <CmdArg>', 4);`
- The system administrator of the remote machine must set up access for the Data Services user. The `.rhosts` and/or the `hosts.equiv` file must have an entry allowing this access.
  - If the remote machine is Windows, the Remote Shell Service must be running on it.
  - If the remote machine is UNIX, the Remote Shell daemon rshd must be running on it.

  Consult your operating system documentation for more information.

Example

The examples below can be used with Windows or UNIX. If you were using the first two examples for UNIX, substitute 'sh', 'csh', 'ksh', 'bash' or 'tcsh' for 'cmd'. Also, the first two examples call 'cmd' rather than the program directly. You need to use 'cmd' (or its equivalent) if either:

- The “command” is a built-in of the shell (e.g., 'DIR' is not a program in Windows)
- Piping (a single '|' in an argument) occurs
- In either Windows or UNIX, the vertical bar symbol sends the output of one command to another command. Only use a vertical bar inside quotes. In Data Services, the double vertical bar symbol (||) concatenates strings. Only use a double vertical bar outside quotes.

Also, remember that the '\', '/' symbols are interchangeable when using Windows. However, only '/' is accepted as a directory separator on UNIX.

```
exec('cmd', 'dir ' || $filename, 8);
exec('cmd', 'x:/bin/program1.exe | x:/bin/postprocess.bat', 4);
exec(SRC.PROGNAME, ARGTBL.ARGS || ' lastArg', 2);
exec('c:\Data Services\bin\clone_and_rename.bat', TBL.FNAME, 1);
exec('C:\Perl5\bin\perl.exe', 'C:\sandbox\stats.pl 20 50 3000', 0);
```

3.6.3.38 extract_from_xml

This function allows you to extract XML data that is stored in one field of a database table into a query's output schema with Data Services' nested relational data model (NRDM) structure. Varchar data types are supported in the input column. You can use the following methods to extract data from clob and long data types.

- Data Services converts a clob data type input to varchar if you select the *Import unsupported data types* as VARCHAR of size option when you create a database datastore connection in the Datastore Editor.
- If your source uses a long data type, use the long_to_varchar function to convert data to varchar.

Syntax

```
extract_from_xml(<xml_column_name>,<schema_dtd_name>,<enable_validation>)
```

Value	Description
xml_column_name	The name of the input column that contains the XML text. The column data type must be varchar.
< schema_dtd_name >	The name of the DTD or XML Schema format that describes the incoming XML text. You must import the metadata for this format into Data Services. Data Services displays the format names in the *Formats* tab of the object library. The input value must be a constant since Data Services needs to know the output schema at design time.
enable_validation	Enable a comparison of the incoming XML data against the format you specify for **<schema_dtd_name>**. The XML data structure and the XML format structure must match. When this option is enabled, the data flow throws an exception if the incoming data is not valid. Enter `1` to validate. Enter `0` to disable the validation option. Validation is useful during development. It provides a more precise error if a problem occurs with the incoming XML.

Where

Once you provide parameter values, the output schema of this function will match that of the XML Format specified. You can select any of the top-level columns or the two columns Data Services generates in the NRDM for output:

- You can select any number of the top-level columns from XML schema. The return type of each column follows that defined in the XML schema.
- AL_ERROR_NUM - Returns an integer which indicates if an error occurred inside the function. A `0` indicates success and any non-zero integer indicates an error.

- AL_ERROR_MSG - Returns an error message if AL_ERROR_NUM is not 0. Otherwise, returns NULL.

Example

```
extract_from_xml(<cust_note_column>,<note_format>,1)
```

Related Information

Designer Guide: Nested Data, Scenario 1 [page 366]

3.6.3.39 file_exists

Checks to see if a given file or directory exists.

Syntax

```
file_exists(<file_path>)
```

Return Value

int

Returns 1 if a file or directory is present on the disk (even if 0 bytes long), 0 otherwise.

Where

<file_path>	The file name and path, relative to where the Job Server is running. It can be an absolute or relative path.

Example

Examples

Invoke sleep for one second when the file temp.msg exists in the directory called "c:".

```
while (file_exists('c:/temp.msg') = 1)
begin
 sleep(1000);
end
```

Set a variable to a file name and use the function to check whether the file exists:

```
$unix_file = '/tmp/t.cpp';
if (file_exists($unix_file)) $type = 'unix';
```

Set a variable based on the value of the function:

```
$i = file_exists('c:/autoexec.bat')
```

3.6.3.40 fiscal_day

Converts a given date into an integer value representing a day in a fiscal year.

Syntax

```
fiscal_day(<start_year_date, in_date>)
```

Return Value

int

Where

<start_year_date>	The first month and day of a fiscal year. Use this format: 'mm.dd'.
<in_date>	The date you want to convert. Use any valid datetime.

Example

Function	Results
`fiscal_day('03.01', '1999.04.20')`	`51`

3.6.3.41 floor

Returns the largest integer value equal to or less than a number.

Syntax

```
floor(<num>)
```

Return value

decimal, double, int, or real

The indicated integer, cast as the same type as the original number, `<num>` .

Where

`<num>`	The source number.

Example

Function	Results
`floor(12.12345)`	`12.00000`
`floor(12)`	`12`
`floor(-12.223)`	`-13.000`

3.6.3.42 gen_row_num_by_group

Generate a column of row IDs for each ID group in the specified column, beginning with integer value 1 and then incremented sequentially by 1. When the group is changed, the value is reset to 1.

Syntax

```
gen_row_num_by_group(expression_list)
```

Return Value

Integer

Where

<expression_list>	A list of one or more comma-separated expressions.

Details

This function groups the rows based on the value of expression(s) in each row in the natural order. It returns the row ID beginning with 1, then increments it sequentially by 1 for each row in the group.

In the example below, the Contract_ID column (shown under Input) shows miscellaneous ID numbers. When the `gen_row_num_by_group` function is applied to the ID column list, the IDs in the Contract_ID column are assigned a new ID number, with the first ID in the list assigned number 1. The ID number on the next row increases by 1 and is assigned the ID number 2. The ID in the following row also increases by an increment of 1, and is assigned the number 3 (as shown in the new Version_Num column under Output in the example).

If the expression(s) corresponds to a column of a table, that column must not be a NRDM or long type column.

This function should not be used with group by clause or any aggregate function.

Example

A typical use case of this function is to assign version numbers, which can become a part of the primary key in the table, as shown below.

Input			
Record	**Contract_ID**	**Revised_by**	**Revision_date**
record 1	1	John	1/1/2005
record 2	1	Mary	1/15/2005
record 3	1	Tim	2/1/2005
record 4	2	Joe	2/24/05

Input			
Record	Contract_ID	Revised_by	Revision_date
record 5	2	Sue	2/30/05

A version = gen_row_num_by_group (Contract_ID) would give an order of Contract_ID to group the contracts:

Output				
Record	Contract_ID	Version_Num	Revised_by	Revision_date
record 1	1	1	John	1/1/2005
record 2	1	2	Mary	1/15/2005
record 3	1	3	Tim	2/1/2005
record 4	2	1	Joe	2/24/05
record 5	2	2	Sue	2/30/05

3.6.3.43 gen_row_num

Returns an integer value beginning with 1 then incremented sequentially by 1 for each additional call. This function can be used to generate a column of row IDs.

Syntax

```
gen_row_num()
```

Return Value

int

Details

Each occurrence of the function in a data flow is a unique instance, resulting in a unique sequence. Two call instances return values independent of each other. The first time an instance of this function is called, the function returns a value of 1. Subsequent calls of the same instance return the previous value incremented by 1 (i.e., 2, 3, 4...).

Each time a data flow is called, all instances are reinitialized, starting at 1.

Example

Use the function in a query's mapping expression to add a column of row IDs to a target.

```
gen_row_num()
```

3.6.3.44 gen_uuid

This function returns a unique identifier.

Syntax

```
gen_uuid()
```

This function takes no parameters.

Return value

Returns a 32-character varchar string. For example, 550e8400e29b41d4a716446655440000.

3.6.3.45 get_domain_description

Returns the description of a value when given the domain name and the value.

Syntax

```
get_domain_description(<domain_name>,<domain_value>)
```

Return value

varchar

The description is returned as a quoted string. If the value is not in the domain, then a NULL is returned.

Where

<domain_name>	Fully qualified domain name, including the database owner if required. For example: **<datastorename>** . **<owner.domain>** **<datastorename>** . . **<domain>**
<domain_value>	The constant value for which you want to return a description.

Example

Function	Results
`get_domain_description('psoft..ACTION','ADL')`	"Additional"

3.6.3.46 get_env

Returns a value for the specified system environment variable.

Syntax

```
get_env('<variable_name'>)
```

Return Value

varchar(255)

Returns the value of the environment variable. Returns NULL if the environment variable is not set. You can use the is_set_env function to determine whether a variable is set.

Where

<variable_name>	The name of the environment variable. The name must be surrounded by single quotes.

Example

Function	Results
`getenv('TMP')`	`C:\Temp`

Related Information

is_set_env [page 1580]

3.6.3.47 get_error_filename

Returns the full path and file name for the error log, which Data Services generates after a job run. Data Services generates log files in the <**`<DS_COMMON_DIR>`**>`\log\`**`<Job Server>`**`\`**`<repository>`** directory. This log starts with 'error_'.

Data Services generates a different set of log files for each batch and real-time (in test mode) job run. Data Services creates only one set of log files during the life of a real-time service.

Syntax

```
get_error_filename()
```

Return Value

varchar

Example

Create a job and add a script with, for example, the following lines.

```
print('Error File Name:');
print(get_error_filename());
```

Returns, for example, the following to the trace log:

```
Error File Name:
d://DI11/log/JS_Ora/repo__riv/
error_12_10_2004_12_06_41_12__8507da25_0b33_4ac1_9b53_fcf1e004c968.txt
```

3.6.3.48 get_file_attribute

Returns the value for a specified attribute of a physical file.

Syntax

```
get_file_attribute(<filename>, <attribute>)
```

Return value

double

If the attribute is `size`.

datetime

If the attribute is either `date_created` or `date_modified`.

Where

`<filename>`	The file name and path relative to the current Job Server. Enter a file name as a relative path or an absolute path.
`<attribute>`	One of the following attributes, which must be in single quotes: date_created, date_modified, or size.

Example

Function	Results
`get_file_attibute('//order', 'date_created')`	'2004:09:15:20:25:00' The format in this example is YYYY:MM:DD:HH24:MM:SS.
`get_file_ attibute(c:\database\or-der,'size')`	'63281' Displays file size in bytes.

Limitations

- The function is not pushed down.

- This function cannot be used in an ABAP data flow.
- For MS Window systems, this function cannot return the create time from FAT formatted disk drives. It works with the NTFS (New Technology File System) format. Most systems use NTFS today because it is more powerful and offers a security advantage over FAT.

3.6.3.49 get_monitor_filename

Returns the full path and file name for the monitor log, which Data Services generates during a job run. Data Services generates log files in the **<DS_COMMON_DIR>**\log**<Job Server>****<repository>** directory. This log starts with 'monitor_'.

Data Services generates a different set of log files for each batch job run and each real-time job run (in test mode). Data Services creates only one set of log files during the life of a real-time service.

Syntax

```
get_monitor_filename()
```

Return Value

varchar

Example

Create a job and add a script with, for example, the following lines.

```
print('Monitor File Name');
print(get_monitor_filename());
```

Returns, for example, the following to the trace log:

```
Monitor File Name
d://DI11/log/JS_Ora/repo__riv/
monitor_12_1_2004_12_06_41_12__8507da25_0b33_4ac1_9b53_fcf1e004c968.txt
```

Note

A monitor log is referred to as a trace log in the Designer.

3.6.3.50 get_trace_filename

Returns the full path and file name for the trace log, which Data Services generates during a job run. Data Services generates log files in the **<DS_COMMON_DIR>**`\log\`**<Job Server>**`\`**<repository>** directory. This log starts with 'trace_'.

Data Services generates a different set of log files for each batch job run and each real-time job run (in test mode). Data Services creates only one set of log files during the life of a real-time service.

Syntax

```
get_trace_filename()
```

Return Value

varchar

Example

Create a job and add a script with, for example, the following lines.

```
print('Trace File Name');
print(get_trace_filename());
```

Returns, for example, the following to the trace log:

```
Trace File Name
d://DI11_97/log/JS_Ora/o920i1__riv/
trace_12_10_2004_12_06_41_12__8507da25_0b33_4ac1_9b53_fcf1e004c968.txt
```

3.6.3.51 greatest

Returns the greatest of the list of one or more expressions.

Syntax

```
greatest(expression_list)
```

Return Value

Data Services uses the first expression to determine the return type. After comparison, the result is converted into the return data type.

Where

`<expression_list>`	A list of one or more comma-separated expressions.

Details

GREATEST returns the greatest of the list of one or more expressions. After comparison, the result is converted into a return data type. Data Services implicitly converts expression in the list to a normalized data type before comparison.

The following rules determine the normalized data type.

- If the return data type is varchar, then all expressions are implicitly normalized to varchar before comparison.
- If the return data type is one of the date data types, then all the expressions in the list are implicitly normalized to that data type before comparison. For example, if the return data type is date, and another data type is 'datetime', then the 'datetime' data type is normalized to 'date' before comparison.
- If the return data type is numeric, then all the expressions are implicitly normalized to the highest precedence numeric expression in the list. For example, greatest (expr1,expr2,expr3,expr4) where expr1 is as integer, expr2 is a decimal (4,2), expr3 is a float, expr4 is a decimal (38,7), then the normalized data type is decimal. All the expressions in the list are converted to decimal data type before comparison. If the normalized data type is decimal, then its precision is the highest precision among all decimal data type expressions. The decimal data type expressions preserve their scale during implicit conversion. When an nteger data type expression is converted to a decimal data type, its scale is 0. When float, double and varchar data types are converted into decimal data types, their scale is 6.

Note

greatest() returns NULL when at least one argument is NULL.

Example

Input				
ID	**GRADE_Q1**	**GRADE_Q2**	**GRADE_Q3**	**GRADE_Q4**
1	'A'	'B'	'B'	'C'
2	'F'	'F'	'E'	'C'
3	'B'	'B'	NULL	NULL

Output	
MAX_GRADE=greatest `(GRADE_Q1,GRADE_Q2,GRADE_Q3,GRADE_Q4)`	
ID	**MAX_GRADE**
1	'C'
2	'F'

Output	
MAX_GRADE=greatest(GRADE_Q1,GRADE_Q2,GRADE_Q3,GRADE_Q4)	
ID	MAX_GRADE
3	NULL

3.6.3.52 host_name

Returns the name of the computer on which the job is executing.

Syntax

```
host_name()
```

Return Value

varchar

Example

```
print('Host Name: [host_name()]');
```

3.6.3.53 ifthenelse

Allows conditional logic in expressions.

Syntax

```
ifthenelse(<condition>, <true_branch>, <false_branch>)
```

Return value

<true_branch> or **<false_branch>**

Returns one of the values provided, based on the result of **<condition>**. The data type of the return value is the data type of the expression in **<true_branch>**. If the data type of **<false_branch>** is not convertible to the data type of `<true_branch>`, Data Services produces an error at validation. If the data types are convertible but don't match, a warning appears at validation.

If `<condition>` compares a varchar value with trailing blanks, the ifthenelse function ignores the trailing blanks.

To compare a NULL value (NULL constant or variable that contains a NULL constant), use the IS NULL or IS NOT NULL operator. If you use the Equal (=) or Not equal to (<>) operator to compare against a NULL value, `<condition>` always evaluates to FALSE.

Where

`<condition>`	An expression that evaluates to TRUE or FALSE.
`<true_branch>`	An expression that the function returns if `<condition>` evaluates to TRUE.
`<false_branch>`	An expression that the function returns if `<condition>` evaluates to FALSE.

To improve performance, Data Services pushes this function to the database. Thus, the database evaluates the IFTHENELSE logic rather than the Data Services engine.

Use this function to apply conditional logic when mapping columns or selecting columns in a query. For more flexible control over conditions in a script, use the IF keyword in the Data Services scripting language.

Example

Function	Results

`ifthenelse (ZIP < 94000, 'SOUTH',`
`'NORTH')`
	If the value in the column `ZIP` is less than 94000, the value returned is `SOUTH`. If `ZIP` is greater than 94000, then the value returned is `NORTH`.

3.6.3.54 index

Returns the index of a given character sequence in a string.

Syntax

```
index(<input_string>, <index_string>, <start>)
```

Return value

int

Specifies the first location of the indicated character sequence.

Where

`<input_string>`	The source string.
`<index_string>`	The character sequence sought in `<input_string>`.
`<start>`	The position where the function starts searching in `<input_string>` for the character sequence contained in `<index_string>`. `<Start>` should be a positive number between 1 and the length of `<input_string>`.

Details

The function searches for the `<index_string>` beginning at the `<start>` position in the input_string. If `<start>` is 0, it is reset to 1; if `<start>` is greater than the number of characters in `<input_string>`, the function returns NULL.

If `<index_string>` is not found in `<input_string>`, the function returns 0. The characters in `<index_string>` must match exactly the sequence of characters in `<input_string>`. The search is case-sensitive.

Example

Function	Results
`index('Accounting Department', 'DEPARTMENT', 1)`	`0`
`index('Accounting Department', 'Department', 1)`	`12`

3.6.3.55 init_cap

Changes the characters in a string to title case. This function converts the first letter of each word to uppercase and the rest of the value to lowercase. It ignores non-alphanumeric characters.

Syntax

```
init_cap(<value,'locale'>)
```

Return value

varchar

The title case string. Words are delimited by white space or characters that are not alphanumeric.

Where

`<value>`	The string to be modified.
`<locale>`	Optional parameter that converts the string to the specified locale. Note ISO 639 language code and ISO 3166 country code formats are supported.

Example

Function	Results
`init_cap('Data Services')`	'Data Services'
`init_cap(StreetAddress)`	Writes the value, for example '1234 west washington school road', in column StreetAddress as '1234 West Washington School Road'.
`Print(Init_cap('have a nice day -hypen +plus _underscore \slash $dollar *star @at   tab mIXedWORd UPPER lower !punctuations 1234digits'));`	Have A Nice Day -Hypen +Plus _Underscore \Slash $Dollar *Star @At Tab Mixedword Upper Lower !Punctuations 1234digits
`init_cap(LastName,'tr')`	The value in column`LastName` will have the first letter capitalized. If there is more than one last name in this column, that string will also have its first letter capitalized. It is also converted to the Turkish locale, using the ISO 639 language code.

Limitations

- The function is pushed down to Oracle databases only.
- You cannot use this function in an ABAP data flow.

Related Information

ISO 639 language list

ISO 3166 Country Code list

3.6.3.56 interval_to_char

Converts an interval value to a string.

Syntax

```
interval_to_char(<input_interval, interval_type>)
```

Return value

varchar

The converted string.

Where

<input_interval>	The value of type `interval` to convert.	
<interval_type>	A string describing the format of the interval. Choose from the following values:	
	D	Days
	H	Hours
	M	Minutes
	S	Seconds

Example

Function	Results
`interval_to_char(start_date - sysdate(), 'd')`	The number of days between the date in the column `start_date` and today's date.
`interval_to_char(start_time - systime(), 'm')`	The number of minutes between the time in the column `start_time` and the current time.

3.6.3.57 is_group_changed

Returns 1 if the group is changed, 0 otherwise.

Syntax

```
is_group_changed(expression)
```

Return Value

Integer

Where

`<expression>`	One or more valid input expressions separated by commas.

Details

This function groups records based on the equal value of the input expressions in parameter1 in the natural order of the input record stream. It returns 1 when the group is changed, 0 otherwise. In the following example, the results show that four of the input groups have changed.

Example

Function	Results
`is_group_changed(state,city)`	`1,0,1,0,0,1,1`

Group ID	State	City	Group change
1	California	Los Angeles	1
2	California	Los Angeles	0
3	California	San Francisco	1
4	California	San Francisco	0
5	California	San Francisco	0
6	Nevada	Reno	1
7	Colorado	Reno	1

3.6.3.58 is_set_env

Verifies if the specified system environment variable is set.

Syntax

```
is_set_env(<variable_name>)
```

Return value

int

Returns 1 if the environment variable is set, otherwise, returns 0.

Where

<variable_name>	The name of the environment variable. The name must be surrounded by single quotes.

Example

Function	Results
`is_set_env('MODE')`	Returns 1 if the MODE variable has already been set; returns 0 if the MODE variable has not been set.

3.6.3.59 is_valid_date

Indicates if an expression can be converted into a valid calendar date value. For example the following will return a negative result:

```
is_valid_date ('01/34/2002', 'mm/dd/yyyy')
```

This expression returns 0 because there is no such date as January 34th.

Syntax

```
is_valid_date(<input_expression, date_format>)
```

Return value

int

- If the expression is not NULL and valid, it returns 1.
- If the expression is not NULL and invalid, it returns 0.
- If the expression is NULL, it returns NULL.

Where

<input_expression>	The expression to be validated. If the expression does not resolve to a value of data type `varchar`, you will see a warning that the value has been converted to a `varchar`.	
<date_format>	The string identifying the date format of the input string. Construct the date format using the following codes and other literal strings or punctuation:	
	DD	2-digit day of the month
	MM	2-digit month

	MONTH	Full name of month
	MON	3-character name of month
	YY	2-digit year
	YYYY	4-digit year

Example

Function	Results
`is_valid_date (Orders.SubmitDate,'mm/dd/yyyy')`	Tests whether the string `Orders.SubmitDate` can be converted to a calendar date with the `mm/dd/yyyy` date format.

Related Information

date [page 1027]

3.6.3.60 is_valid_datetime

Indicates if an expression can be converted into valid calendar date and time values. For example the following will return a negative result:

```
is_valid_datetime ('01/14/2002 26:56:09', 'mm/dd/yyyy hh24:mi:ss')
```

This expression returns 0 because there is no such hour as "26", even on the 24 hour clock.

Syntax

```
is_valid_datetime(<input_expression, datetime_format>)
```

Return value

int

- If the expression is not NULL and valid, it returns 1.
- If the expression is not NULL and invalid, it returns 0.
- If the expression is NULL, it returns NULL.

Where

<input_expression>	The expression to be validated.	
<datetime_format>	The string identifying the datetime format of the input expression. Construct the datetime format using the following codes and other literal strings or punctuation:	
	DD	2-digit day of the month
	MM	2-digit month
	MONTH	Full name of month
	MON	3-character name of month
	YY	2-digit year
	YYYY	4-digit year
	HH24	2-digit hour of the day (00-23)
	MI	2-digit minute (00-59)
	SS	2-digit second (00-59)

Example

Function	Results
`is_valid_datetime (Orders.Received,'mm/dd/yyyy hh24:mi:ss')`	Tests whether the string `Orders.Received` can be converted to the `mm/dd/yyyy hh24:mi:ss` datetime format.

Related Information

datetime [page 1029]

3.6.3.61 is_valid_decimal

Indicates if an expression can be converted into a valid decimal value.

Syntax

```
is_valid_decimal(<input_expression, decimal_format>)
```

Return value

int

- If the expression is not NULL and valid, it returns 1.
- If the expression is not NULL and invalid, it returns 0.
- If the expression is NULL, it returns NULL.

Where

<input_expression>	The expression to be validated.
<decimal_format>	A string indicating the decimal format of the input expression. Use pound characters (#) to indicate digits and a decimal indicator. If necessary, include commas as thousands indicators. For example, to specify a decimal format for numbers smaller than 1 million with 2 decimal digits, use the following string: `'#,###,###.##'`. To indicate a negative decimal number, add a minus "-" sign at the beginning or end of this value. For example, to test if the stock price difference can be converted to decimal format, use the following function: `is_valid_decimal (Stocks.Price_difference, '-###.##')`

Example

Function	Results
`is_valid_decimal (Orders.Price,'###,###.##')`	Tests whether the string `Orders.Price` can be converted to decimal format.

3.6.3.62 is_valid_double

Indicates if an expression can be converted into a valid double value.

Syntax

```
is_valid_double(<input_expression, double_format>)
```

Return value

int

- If the expression is not NULL and valid, it returns 1.
- If the expression is not NULL and invalid, it returns 0.
- If the expression is NULL, it returns NULL.

Where

<input_expression>	The expression to be validated.
<double_format>	A string indicating the double format of the input expression. Use pound characters (#) to indicate digits and a decimal indicator. If necessary, include commas as thousands indicators. For example, to specify a double format for numbers smaller than 1 million with 2 decimal digits, use the following string: `#,###,###.##'`

Example

Function	Results
`is_valid_double (Product.Weight,'###.###')`	Tests whether the string `Product.Weight` can be converted to double format.

3.6.3.63 is_valid_int

Indicates if an expression can be converted into a valid integer value.

Syntax

```
is_valid_int(<input_expression, int_format>)
```

Return value

int

- If the expression is not NULL and valid, it returns 1.
- If the expression is not NULL and invalid, it returns 0.
- If the expression is NULL, it returns NULL.

Where

<input_expression>	The expression to be validated.
<int_format>	The format specifying the thousands separator of the input expression. For example, to specify an integer format, use the following string: #.###.###'. Valid separators include the period (.) and the comma (,). However, you can only use one valid separator type in a format. Separator defaults to the comma (,) when none is specified.

Example

Function	Results
`is_valid_int (QuarterResults.Volume,'###.###')`	Tests whether the string `QuarterResults.Volume` can be converted to the `###.###` integer format.

3.6.3.64 is_valid_real

Indicates if an expression can be converted into a valid real value.

Syntax

```
is_valid_real(<input_expression>, <real_format>)
```

Return value

int

- If the expression is not NULL and valid, it returns 1.
- If the expression is not NULL and invalid, it returns 0.
- If the expression is NULL, it returns NULL.

Where

<input_expression>	The expression to be validated.
<real_format>	A string indicating the real format of the input expression. Use pound characters (#) to indicate digits and a decimal indicator. For example, to specify a real format for numbers smaller than 1 million with 2 decimal digits, use the following string: `'#,###,###.##'`.

Example

Function	Results
`is_valid_real (QuarterResults.Mean,'#,###.#####')`	Tests whether the string `QuarterResults.Mean` can be converted to real format.

3.6.3.65 is_valid_time

Indicates if an expression can be converted into a valid time value.

Syntax

```
is_valid_time(<input_expression, time_format>)
```

Return value

int

- If the expression is not NULL and valid, it returns 1.
- If the expression is not NULL and invalid, it returns 0.
- If the expression is NULL, it returns NULL.

Where

<input_expression>	The expression to be validated.

<time_format>	The string identifying the time format of the input expression. Construct the time format using the following codes and other literal strings or punctuation:	
	HH24 MI SS	2-digit hour of the day (00-23) 2-digit minute (00-59) 2-digit second (00-59)

Example

Function	Results
`is_valid_time (Orders.ReceivedTime,'hh24:mi:ss')`	Tests whether the string `Orders.ReceivedTime` can be converted to the `hh24:mi:ss` datetime format.

3.6.3.66 isempty

Indicates if a nested table contains data.

Syntax

```
isempty(<table_name>)
```

Return value

int

The result of the content test: returns 1 if the table does not contain data; returns 0 if the table does contain data.

Where

<table_name>	The fully qualified name of the nested table to test. A fully qualified name contains the parent table names up to the top level of the table in the current context. If you only specify a table name, Data Services looks for the table among the tables available through the

	FROM clause of the current context. If you specify a partially qualified table name (only part of the table hierarchy), Data Services looks for the table among the tables available in the FROM clause of the context indicated by the partial qualification.

When performing operations on hierarchical data, the isempty function allows you to exclude rows in a higher-level table based on whether a lower-level table contains data.

Data Services determines that a nested table is empty when the table contains no rows. If the nested table contains even one row with null values in all columns, the isempty function indicates that the table has content. If the nested table is empty except for another nested table, and the second nested table does contain data, then the first nested table is not empty.

Example

You can use the isempty function to determine if there are line items associated with a sales order. For example, if the sales order is the input data set to a Query transform and you want the query to exclude orders without line items, include the following expression in the WHERE clause of the top-level context of the query:

```
isempty (order_table.line_items_table)
```

3.6.3.67 isweekend

Indicates if a date corresponds to Saturday or Sunday.

Syntax

```
isweekend(<date1>)
```

Return value

int

The result of the date test: returns 1 if the date is a Saturday or Sunday; returns 0 if not.

Where

<date1>	The value of type `date` or `datetime` to test.

Example

Function	Results
`isweekend(hire_date)`	Tests whether the date in `hire_date` is a Saturday or Sunday.
`isweekend(SYSDATE)`	Tests whether today is a Saturday or Sunday.

3.6.3.68 job_name

Returns the name of the job in which the call to this function exists.

Syntax

```
job_name()
```

Return Value

varchar

Example

```
print('Starting execution of Job: [job_name()] as user: [system_user_name()]');
```

3.6.3.69 Job_Run_ID

Syntax

Retrieves the job run ID for the current job execution.

Syntax

```
job_run_id()
```

Return value

Varchar (20)

3.6.3.70 julian

Converts a date to its integer Julian value, the number of days between the start of the Julian calendar and the date.

Syntax

```
julian(<date1>)
```

Return value

int: The Julian representation of the date.

Where

`<date1>`	The source value of type `date` or `datetime`.

Example

Function	Results
`julian(to_date('Apr 19, 1997', 'mon dd, yyyy'))`	`729436`

3.6.3.71 julian_to_date

Converts a Julian value to a date.

Syntax

```
julian_to_date(<input_julian>)
```

Return value

date

The date that corresponds to the input Julian value.

Where

`<input_julian>`	An integer representing the Julian value to be converted.

Example

Function	Results
`julian_to_date(Julian_Date)`	Converts the number indicated by `Julian_Date` to its date value.

3.6.3.72 key_generation

Generates the next value in a series, after determining the last value in the series.

The key_generation function determines the maximum existing key value in a given column in the table in the database manager and uses that value as a starting point to generate key values for the target schema.

Note

The key_generation function is not supported when using SAP tables as readers.

Syntax

```
key_generation (<table>, <key_column>, <key_increment>)
```

Return value

int

The column value found to meet the function requirements.

Where

<table>	The full path name of the file or full database specification of the table in which the **<key_column>** is located. Enclose this value in single quotation marks.
<key_column>	A column with existing keys from which this function determines the largest existing key value. Enclose this value in single quotation marks.
<key_increment>	The integer increment used between key values this function generates.

Example

Function	Results
`key_generation('target_ds.dbo.sales','order_number',1)`	Looks for the last key value in the `order_number` column in the sales database and returns the largest value plus one.

3.6.3.73 last_date

Returns the last date of the month for a given date.

Syntax

```
last_date(<in_date>)
```

Return Value

date

Where

<in_date>	The date for which the last date of the month is to be calculated.

Example

Function	Returns
`last_date('1990.10.01')`	`'1990.10.31'`

3.6.3.74 least

Returns the least of the list of one or more expressions.

Syntax

```
least(expression_list)
```

Return Value

Data Services uses the first expression to determine the return type. After comparison, the result is converted into the return data type.

Where

<expression_list>	A list of one or more comma-separated expressions.

Details

least returns the least of the list of one or more expressions. After comparison, the result is converted into a return data type. Data Services implicitly converts expressions in the list to a normalized data type before comparison.

The following rules determine the normalized data type:

1. If the return data type is varchar, then all expressions are implicitly normalized to varchar before comparison.
2. If the return data type is one of the date data types, then all the expressions in the list are implicitly normalized to that data type before comparison. For example, if the return data type is date, and another data type is 'datetime', then the 'datetime' data type is normalized to 'date' before comparison.
3. If the return data type is numeric, then all the expressions are implicitly normalized to the lowest precedence numeric expression in the list. For example, least(expr1,expr2,expr3,expr4) where expr1 is aa integer, expr2 is

a decimal (4,2), expr3 is a float, expr4 is a decimal (38,7), then the normalized data type is decimal. All the expressions in the list are converted to decimal data type before comparison. If the normalized data type is decimal, then its precision is the lowest precision among all decimal data type expressions. The decimal data type expressions preserve their scale during implicit conversion. When an integer data type expression is converted to a decimal data type, its scale is 0. When float, double and varchar data types are converted into decimal data types, their scale is 6.

Note

least() returns NULL when at least one argument is NULL.

Example

		Input		
ID	GRADE_Q1	GRADE_Q1	GRADE_Q3	GRADE_Q4
1	'A'	'B'	'B'	'C'
2	'F'	'F'	'E'	'C'
3	'B'	'B'	NULL	NULL

	Output	
	MIN_GRADE=least (GRADE_Q1,GRADE_Q2,GRADE_Q3,GRADE_Q4)	
ID	MAX_GRADE	MIN_GRADE
1	'C'	'A'
2	'F'	'C'
3	NULL	NULL

3.6.3.75 length

Returns the number of characters in a given string.

Syntax

```
length(<value>)
```

Return value

integer

The number of characters in `<value>`.

Where

`<value>`	A string indicating the column name, variable, or other element whose length is calculated.

Example

In the *Mapping* box of a query, you can use the length function to return the number of characters in each row of a column. For example, with the OUTPUT field selected in the target schema of a query, entering the following statement in the *Mapping* box:

```
length(dal_emp.ename)
```

produces the following results:

Source column (dal_emp.ename)	Target column (output)
`jones`	`5`
`nguyen`	`6`
`tanaka`	`6`

3.6.3.76 literal

Returns an input constant expression without interpolation. Data Services normally does not use variable interpolation on constants. However, if you pass in a variable as a constant expression, Data Services automatically uses variable interpolation, replacing special characters.

This is an issue with the Match_ pattern and Match_regex functions because they require these special characters. If your `pattern_string` or `regular_expresion_pattern` parameter in these functions is a constant, you may want to disable interpolation. If so, use the Literal function.

If, for example, you want to match $my_pattern with the pattern 'PART[123]'.

If you coded it simply as:

`$my_pattern = 'PART[123]'; match_pattern(product,$my_pattern);` the interpolation would actually change the pattern being matched to 'PART123', but if you code it as:

```
$my_pattern = literal ('PART[123]');
```

```
match_pattern(product, $my_pattern);
```

then it could return 1, because the pattern would remain 'PART[123]'.

Alternatively, if you do not want to use a variable, you can code it as:

```
match_pattern (product,'PART[123]');
```

because in this case no interpolation is done on the constant 'PART[123].

There is no runtime cost for the Literal function. Data Services substitutes the constant expression at compile time.

Syntax

```
literal(<input>)
```

Return value

Same as that of the value given for the input parameter but without interpolation.

Where

<input>	A constant expression of any data type.

Example

If you want to match only PART1 or PART2 or PART3 using the Match_pattern function, you must assign a pattern to a variable without interpolation. Use the Literal function in the following type of expression:

```
$pattern = literal('PART[123]');
```

For example, if you do not use the literal function, the value assigned to $my_pattern in the following sample is 'PART123' because Data Services automatically removes square brackets during interpolation.

```
$my_pattern='PART[123]';
print($my_pattern);
if (match_pattern('PART1',$my_pattern) <> 0)
 print('Matched');

else
print('Not Matched');
```

To disable interpolation, use the Literal function. The following example returns the result you want.

```
$my_pattern=LITERAL('PART[123]');
print($my_pattern);
if (match_pattern('PART1',$my_pattern) <> 0)
print('Matched');

else
print('Not Matched');
```

3.6.3.77 ln

Returns the natural logarithm of the given numeric expression.

Syntax

```
ln(numeric_expression)
```

Return Value

Float

Where

<numeric_expression>	Any numeric expression.

Details

Returns the natural logarithm of the given numeric expression. Return value is NULL if input is negative.

Example

Function	Results
`ln(5.436563656918)`	`1.693147`

3.6.3.78 load_to_xml

This function converts NRDM into XML and places it in a single column during a data load.

If the function fails due to an error when trying to produce the XML output, Data Services returns NULL for scalar columns (that you select for output) and an empty nested table for NRDM columns.

Syntax

```
load_to_xml (nested_table_name, schema_dtd_name, enable_validaton, xml_header,
replace_string_nulls, is_top_level_element, max_size)
```

Where

`nested_table_name`	The name of the NRDM table that you want to convert into an XML format.
`schema_dtd_name`	The name of the DTD or XML Schema that you want Data Services to use to format the XML text in the output column. Be sure to match the structure of the nested table to that provided by the DTD or XML Schema (or match the structure of its only child). Otherwise, Data Services will not produce output. See the `is_top_level_element` for an example of specifying an only child. Note You can generate an XML Schema or DTD from an NRDM schema using Data Services.
`enable_validation`	Enter `1` to validate. Enter `0` to disable the validation parameter. Validates that Data Services generates XML that matches the XML format you specify for schema_dtd_name.
`xml_header`	If the value specified is NULL, the header of the output XML has UTF-8 in the header. The default header generated is: <?xml version="1.0" encoding="UTF-8"?> If this value is not null, Data Services will replace the default XML header with the one you provide. Make sure the XML_header you provide matches the encoding of the target datastore where you will store the XML data.
`replace_null_string`	If this value is not null, Data Services replaces NULL values with the specified string.
`is_top_level_element`	Enter `1` if the input column representing the NRDM table matches the root element of the given DTD or XML Schema. Enter `0`, if the input column should match the only child of the root element of the DTD or XML Schema. Examples: • Imagine that an XML reader creates an NRDM structure with the root element named PO and an XML Schema po.xsd, which defines a root element also called PO . In this case, when you select the input column name

	PO as the `nested_table_name`, it matches the root element of the schema, so set the value of `is_top_level_element` to `1`. • By contrast, imagine that a database table called employees has rows which each contain information about one employee. Also, you are using an XML Schema called employees.xsd. This schema defines a root element called allEmployees and defines a single element called employee (with unbounded occurrence). In Data Services, the employee table exists in an NRDM structure with employee as the root name. If you enter employee is as the `nested_table_name`, because it does not match the root element of the schema allEmployees, set the value for `is_top_level_element` to `0`. The function returns data with allEmployees as the root element and the number of employee elements under it to match the number of rows in the input employees table.
`max_size`	The expected maximum size of the generated XML.

Example

```
load_to_xml(nested_table_name, billing_address_schema, 0, '<?xml version="1.0"
encoding="UTF-8"?>', NULL, 1, 4000)
```

Related Information

Designer Guide: Nested Data, XML extraction and parsing for columns [page 366]
Designer Guide: Nested Data, Generating DTDs and XML Schemas from an NRDM schema [page 355]

3.6.3.79 log

Returns the base-10 logarithm of the given numeric expression.

Syntax

```
log(num)
```

Return Value

Float

Where

`<num>`	The number for which you want a base- 10 logarithm returned.

Details

Returns the base-10 logarithm of the given numeric expression. Return values is NULL if input is negative.

Example

Function	Results
`log(100.000)`	`2.000000`

3.6.3.80 long_to_varchar

Converts a data type value of a given column from long to varchar.

Syntax

```
long_to_varchar(<column_name>, <max_size, start_position>)
```

Return value

varchar

Where

`<column_name>`	The name of the column containing the long data type.
`<max_size>`	The maximum size for the converted data in the table column.
`<start_position>`	Starting character position from which data is converted. The default start position is 1. A negative num-

(Optional)	ber indicates that the starting position is counted backwards from the last character.

Example

long_to_varchar`(content_column, 4000)`

long_to_varchar`(content_column, 4000, -5000)`

Related Information

Designer Guide: XML extraction and parsing for columns, Scenario 1 [page 366]

3.6.3.81 lookup

Retrieves a value in a table or file based on the values in a different source table or file.

Note

You cannot use this function with a J. D. Edwards datastore. Use the lookup_ext function instead.

Syntax

```
lookup (<lookup_table>, <result_column>, <default_value>, <cache_spec>,
<compare_column>, <expression>)
```

Return value

any type

The value in the **`<lookup_table>`** that meets the lookup requirements. The return type is the same as **`<result_column>`**.

Where

`<lookup_table>`	The table or file that contains the result or value you are looking up (**`<result_column>`**). The **`<compare_column>`** is also located in this table. Use a fully qualified table name, including the datastore, owner, and table name. For example:

	`oracle_ds.TIGER.sales` You might need to put the owner in quotes, particularly if you use lower case letters.
`<result_column>`	The column containing the values you want to retrieve. This column is in the **`<lookup_table>`**.
`<default_value>`	The value returned when there is no matching row in the **`<lookup_table>`**.
`<cache_spec>`	The caching method the lookup operation uses. List within single quotes. There are three possible settings: NO_CACHE: Reads values from the **`<lookup_table>`** for every row without caching values. PRE_LOAD_CACHE: Loads the **`<result_column>`** and **`<compare_column>`** into memory after applying filters but before executing the function. Select this option if the number of rows in the table is small or you expect to access a high percentage of the table values. DEMAND_LOAD_CACHE: Loads **`<result_column>`** and **`<compare_column>`** values into memory as the function identifies them. Select this option if the number of rows in the table is large and you expect to access a low percentage of the table values frequently. Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.
`<compare_column>`	The column in the **`<lookup_table>`** that the function uses to find a matching row. When the function reads a varchar column in the **`<lookup_table>`**, it does not trim trailing blanks.
`<expression>`	The value that the function searches for in the **`<compare_column>`**. This can be a simple column reference, such as a column found in both a source and the **`<lookup_table>`**. This can also be a complex expression given in terms of constants and input column references. When **`<expression>`** refers to a unique source column, you do not need to include a table name qualifier. If **`<expression>`** is from another table or is not unique among the source columns, you need a table name qualifier. If **`<expression>`** is an empty string, the function searches for a zero-length varchar value in the **`<compare_column>`**. The function ignores trailing blanks in comparisons of **`<expression >`** and values in **`< compare_column>`**.

ℹ Note

You can specify more than one `<compare_column>` and `<expression>` pair—simply add additional pairs at the end of the function statement. The values must match for all specified pairs in order for the lookup function to find a matching row.

The lookup function uses a value you provide (`<expression>`) to find a corresponding value in a file or different table. Specifically, the function searches for the row in the `<lookup_table>` where the value in the `<compare_column>` matches the value in `<expression>` . The function returns the `<result_column>` value from this matching row.

For example, if your source schema uses a customer ID to identify each row, but you want the customer name in your target schema, you can use the lookup function to return the customer name given the customer ID.

In SQL terms, the lookup function evaluates `<expression>` for each row, then executes the following command:

```
SELECT <result_column>
    FROM <lookup_table>
    WHERE <compare_column> = <expression>
```

The value returned by this SELECT statement is the result of the lookup function for the row.

You can specify multiple `<compare_column>` and `<expression>` pairs to uniquely identify the `<result_column>` value. However, the function wizard only provides fields for one pair; add extra `<compare_column>` and `<expression>` pairs to the output that the wizard generates.

When there are no matching rows in the `<lookup_table>`, the lookup function returns the `<default_value>`. When multiple matching rows exist in the `<lookup_table>`, the row returned is based on whether the lookup table is a standard RDBMS table, an SAP application table, or a flat file:

- For standard RDBMS tables, the lookup function will find the matching row with the maximum value in the `<result_column>` and return that value.
- For SAP application tables or flat files, the lookup function randomly selects a matching row and returns the value in the `<result_column>` for that row.

ℹ Note

To avoid random row selection when the `<lookup_table>` is an SAP application table or a flat file, it is recommended that you use the lookup_seq() function.

To enhance performance, you can configure the lookup function to hold the values from the `<lookup_table>` in memory. To do so, use the `<cache_spec>` setting. The optimal setting depends on the number of rows the function must read, the number of rows in the table, and the available memory.

Example

You can use the lookup function to return a text value given a numerical identifier. For example, suppose you have a source table containing a numerical identifier, such as an employee number, and you want to use the employee's name in your target.

You can use the lookup function to return the employee name based on the employee number. The lookup function uses a third table that translates the values from the source table to the desired values in the target table.

To produce the desired target column, select the column that you want to look up in the target schema. Click the *Functions* button, located over the *Mapping* text box. The function wizard opens. Under *Function categories*, select *Miscellaneous_Function*, then under *Function name*, select *lookup*. Click *Next*. The Define Input Parameters window opens.

Enter the function parameters as follows:

Option	Value
Lookup table	ODS_DS.SSB.EMPLOYEE
Result column	LastName
Default value	'NoLastName'
Cache spec	'No_CACHE'
Compare column	EmployID
Expression	EmployID

The **<expression>** value refers to a column in the source file or table and therefore does not require qualification with a table name. If this **<expression>** was from another table or was not unique among the source columns, it would require a table name qualifier.

The function wizard automatically produces the mapping text.

```
lookup(SQL_rbh.DBO.ODS_EMPLOYEE, LastName, 'NoLastName', 'NO_CACHE',
    EMP_ID, EMP_ID)
```

You can create a lookup function with two **<expression>** and **<compare_column>** pairs:

```
lookup(sap_ds..VBUP, GBSTA, 'none', 'NO_CACHE', VBELN, VBAK.VBELN, POSNR,
VBAP.POSNR)
```

This function returns the value from the GBSTA column in the VBUP table that corresponds to the VBELN value in the VBAK table and the POSNR value in the VBAP table. When no corresponding value is found, the function returns "none."

3.6.3.82 lookup_ext

Syntax

The following syntax includes line breaks for clarity.

Note

This function has a graphical editor.

```
lookup_ext(
[<lookup_table,cache_spec,return_policy>],
[<return_column_list>],
[<default_value_list>],
[<condition_list>],
[<orderby_column_list>],
[<output_variable_list>],
[<sql_override>]
set ("run_as_separate_process"='yes'),
("output_cols_info"='<?xml version="1.0"
encoding="UTF-8"?>
<output_cols_info>
<col index="1" expression="yes"/>
</output_cols_info>')
)
```

Return value

any type

The return type is the first lookup column in **`<return_column_list>`**.

Where

`<lookup_table>`	The table, file, or memory datastore that contains the result(s) or value(s) you are looking up (**`<result_column_list>`**). If the **`<lookup_table>`** is a database table, use the **`<datastore.owner.table>`** format. For example:
	`ERP_ds.OWNER.EMPLOYEES`
	If the **`<lookup_table>`** is a flat file, use the **`<file_ds.filename>`** format. For example:
	`delim."c:/temp/employees"`
	To substitute a variable for a file name, replace the data inside the double quotes, for example `delim."$employees"`. The variable used to store a file name can be a local or global variable or a parameter passed to a work flow or a data flow. If the cache specification is NO_CACHE, SAP Data Services can pass in a different file name each time it calls lookup_ext. For example, you can call lookup_ext in a WHILE loop and assign a different file name to the variable passed as the lookup file. If the cache specification is PRE_LOAD_CACHE or DEMAND_LOAD_CACHE, only the first file name passed is used. The software ignores all file names passed during subsequent calls.
	If the **`<lookup_table>`** is a memory datastore table, use the **`<memory_ds.table>`** format. For example:
	`mem_ds..employees`

`<cache_spec>`	The caching method the lookup_ext operation uses. You can select one of the following cache specifications:

	• NO_CACHE—Reads values from the `<lookup_table>` for every row without caching values. • PRE_LOAD_CACHE—Loads the `<return_column_list>`, `<compare_column>` (see `<condition_list>`), and `<orderby_column_list>` into memory after applying constant filters and before executing the function. Select this option if the number of rows in the table is small or you expect to access a high percentage of the table values. • DEMAND_LOAD_CACHE—Loads `<return_column_list>`, `<compare_column>` (see `<condition_list>`), and `<orderby_column_list>` into memory as the function identifies them. Select this option if the number of rows in the table is large and you expect to frequently access a low percentage of table values. Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.
`<return_policy>`	Use `<return_policy>` when you expect multiple rows and want output data from one of the selected rows. This optional parameter specifies whether the return columns should be obtained from the smallest or the largest row based on values in the order by columns. The value can be MAX (default), MAX-NS, MIN, or MIN-NS. MAX-NS and MIN-NS allow the lookup_ext function to treat NULL as the smallest value instead of the largest value.
`<return_column_list>`	A comma-separated list containing the names of output columns in the `<lookup_table>`. For a given output column in the lookup table, select the *Expression?* check box in the lookup_ext editor when some of the data is in the form of expressions. If the Expression check box is selected and the data begins with an equals sign (=), the software evaluates the data as an expression and returns the result. Otherwise, it returns the column value.
`<default_value_list>`	A comma-separated list containing the default expressions for the output columns. When no rows match the lookup condition, the default values are returned for the output column. Each default expression type must be compatible with the corresponding output column type such that if the types are not exactly the same, automatic conversion is still possible. If `<default_value_list >` is empty or has fewer expressions than the number of output columns, NULL is the default. You cannot have more default expressions than the number of output columns.
`<condition_list>`	A list of triplets that specify lookup conditions. Each set in a triplet contains `<compare_column>`, `<compare_operator>` (<,<=,>, >=, =, !=, IS, IS NOT, ~), and `<compare_expression>`. The `<compare_column>` is from the `<lookup_table>`. It is compared against `<compare_expression>` to compute the output row.

	The **<compare_expression>** is written in terms of constants, variables, and columns in the calling data flow or scripts. While it cannot contain column reference from the **<lookup_table>**, it can be a simple constant, variable, or column reference or a complex expression involving arithmetic operations and function calls. Use compare operators IS and IS NOT to examine **<compare_column>** against the NULL constant. When you use IS or IS NOT as the compare operator, **<compare_expression>** must contain the NULL constant. When you use other compare operators against a **<compare_expression>** containing a NULL, the lookup condition return value will always return FALSE. Use the compare operator ~ to indicate that the column from the lookup table contains a pattern. The required pattern tags in the lookup table are: ● `mp(`**<pattern>**`)`—Indicates the match_pattern type of pattern syntax ● `mr(`**<pattern>**`)`—Indicates the match_regex type of pattern syntax ● `ms(`**<pattern>**`)`—Indicates the match_simple type of pattern syntax If **<compare_expression>** is an empty string, the function looks up a zero-length varchar value in the lookup table. The function ignores trailing blanks in the **<compare_expression>**. If you create more than one set of triplets, all triplets are implicitly joined with the AND operator to compute the final lookup condition. Example: `[c1, '=', 10,c2,'<',query.a,c3,'>=',lower(query.name)]`
<orderby_column_list>	A comma-separated list of column names from the **<lookup_table>**. Working together with **<return_policy>**, the **<orderby_column_list>** is used to determine which row to return the output when more than one row satisfies the lookup condition. When multiple rows occur, the list of rows is sorted based on columns from the **<orderby_column_list>** and choosing a row to return using the MIN/MAX **<return_policy>**. The **<orderby_column_list>** is optional. If you leave it blank, the orderby columns match the output columns. Examples: [c1,c3,c4] — Sorts the rows using column values in c1, c3, c4. [ ] — Indicated an empty list, which is a placeholder for specifying subsequent parameters.
<output_variable_list>	A comma-separated list of output variables. When more than one output column is specified in the function call, the output variables are used to receive output returns. Variables and output columns are matched by position.

	This parameter is optional unless more than one output column appears in the **`<return_column_list>`**. In the case of more than one output column, output variables must be equal in number to output columns. To enable conversion, the variable data type must be compatible with the corresponding output column. You do not need to specify output variables if the function is called using the function wizard to map output columns in the query window. Example: [$a,$b,$c]

`<sql_override>`	This parameter, available as a button called *Custom SQL* in the function wizard, must contain a valid, single-quoted SQL SELECT statement or a $variable of type VARCHAR to populate the lookup cache when the cache specification is PRE_LOAD_CACHE. This parameter replaces the SQL SELECT statement generated internally by the function for populating the cache. The SELECT statement must select at least those columns contained in **`<return_column_list>`**, **`<condition_list>`**, and **`<orderby_column_list>`**.
	Any valid SQL SELECT statement is permitted and may contain references to other tables besides the **`<lookup_table>`** to specify inner and outer joins. This parameter can only be specified when the **`<lookup_table>`** is a database table.
	If you specify this parameter with the cache specification of NO_CACHE, the software executes the sql_override query each time the function is called.
	If you specify this parameter with the cache specification of PRE_LOAD_CACHE, only the first sql_override query is executed to populate the lookup cache. All subsequent SQL statements are ignored after the lookup cache is built.
	If this parameter is specified when the cache specification is DEMAND_LOAD_CACHE, the caching mode will be converted to PRE_LOAD_CACHE and behave as if the PRE_LOAD_CACHE mode had been specified.
	EXAMPLE:
	`['select out1,`
	`                                    out2,compare1,compare2,orderby1,orderby2 from lookuptbl,othertbl where c1=10`
	`                                    and lookuptbl.c2=othertbl.c2']`
	Also, when you use **`<sql_override>`** in NO_CACHE mode, lookup_ext will dynamically execute the SQL if you pass in a dynamic SQL statement in the form of a variable.

The SET options include:

run_as_separate_process	Select (set to `yes`) to run each operation as a separate process (sub data flow) that uses separate resources (memory and computer) to improve performance and throughput. The default is `no`. `SET ("run_as_separate_process"='yes')`

output_cols_info	Identifies whether an output column contains an expression (when the *Expression* check box is selected). The default is `no`. The syntax is as follows:
	`SET ("output_cols_info"='<?xml version="1.0"`
	`encoding="UTF-8"?>`
	`<output_cols_info>`
	`<col index="1" expression="yes"/>`
	`</output_cols_info>')`
	The first column begins with index value of 1.

Optimizing database push-down

For best performance, the lookup_ext function can be pushed down to the database when the following conditions are met:

- The lookup_ext function is used in the column mapping, output schema, or `SELECT WHERE` clause of a Query transform.
- The **`<lookup_table>`** is a database table from the same datastore or a linked datastore as the reader.
- The **`<cache_spec>`** is set to `NO_CACHE`.
- The **`<return_policy>`** is set to either `MAX` or `MIN`.
- All conditions used in the **`<condition_list>`** are database expressions.
- Only the equals operator (=) are used in the lookup **`<condition_list>`**.
- The **`<run_as_separate_process>`** SET option is set to `no`.
- For lookups with multiple-result column values, the database must support the rank (or equivalent) function.

Note

For SAP HANA, MySQL, SAP Sybase ASE, and Informix databases, no analytic function support is available. As a result, push-down is supported in all cases for single-result columns, and multiple-result columns only for primary keys.

Limitations

- You cannot call the lookup_ext function from an ABAP data flow; use the lookup() function as an alternative.
- When calling lookup_ext in script objects, jobs, or work flows, the caching mode is always NO_CACHE because the software cannot determine when to release the cache after executing the function.
- In all parameters of lookup_ext, you can refer to any supported data type except the long, blob, and NRDM data types. Therefore, you cannot look up a long column or specify a column or expression of the long data type in **`<default_value_list>`**, **`<orderby_column_list>`**, or **`<condition_list>`**.
- If an optional parameter is missing, an empty placeholder ([]) must occupy that position if other optional parameters that follow the missing parameter are specified in the function call.
- It's recommended that for best performance you use the equals operator to specify the lookup condition. If the caching mode is NO_CACHE and the lookup_ext is against a database table, the underlying DBMS typically has fast access methods such as an index to retrieve data based on an indexed key. When the

caching method is PRE_LOAD_CACHE, using the equals comparison will result in more efficient memory lookup than any other comparison operators.
If the caching method is PRE_LOAD_CACHE, any lookup condition involving a constant expression will be pushed down to the database, resulting in a smaller lookup cache than the current lookup.

- Pattern evaluation uses virtual memory and is not included as part of pageable cache when the lookup table is cached. So if the lookup table has a lot of patterns, then the data flow could run out of memory. In those cases, select the *Run as a separate process* check box. This limitation also applies when using expressions in output columns.

Related Information

Designer Guide: Data Flows, Lookup tables and the lookup_ext function [page 293]
match_pattern [page 1622]
match_regex [page 1625]
match_simple [page 1630]
Performance Optimization Guide: Run as a separate process option [page 2153]

3.6.3.83 lookup_seq

Retrieves a value in a table or file based on the values in a different source table or file and a particular sequence value.

Syntax

```
lookup_seq (<lookup_table>, <result_column>, <default_value>, <sequence_column>,
<sequence_expression>, <compare_column>, <expression>)
```

Return value

any type

The value in the **<lookup_table>** that meets the lookup_seq requirements. The return type is the same as **<result_column>**.

Where

<lookup_table>	The table or file that contains the result or value you are looking up (**<result_column>**). The **<sequence_column>** and **<compare_column>** are also

	located in this table. Use a fully qualified table name, including the datastore, owner, and table name. For example: ERP_ds.OWNER.EMPLOYEES The `<lookup_table>` is cached automatically for the operation of the function.
`<result_column>`	The column containing the values you want to retrieve. This column is in the `<lookup_table>`. When the column contains varchar values, the function does not trim trailing blanks.
`<default_value>`	The value returned when there is no matching row in the `<lookup_table>`.
`<sequence_column>`	The column in `<lookup_table>` that indicates the sequence of the row. This column often contains a date that indicates when new values were added to the row. For example, in some source tables, `<sequence_column>` is the `EFFDT` column, which indicates when the data in the row became effective.
`<sequence_><expression>`	The value the function searches for in the `<sequence_column>` to find a matching row. For example, if you are looking up values from a slowly changing dimension table and are interested in only those rows in which the data is current as of today, you could use the return value from the sysdate function for `<sequence_expression>`. If `<sequence_expression>` is an empty string, the function looks up a zero-length varchar value in the lookup table.
`<compare_column>`	The column in the `<lookup_table>` that the function uses to find a matching row.
`<expression>`	The value that the function searches for in the `<compare_column>`. This can be a simple column reference, such as a column found in both a source and the `<lookup_table>`. This can also be a complex expression given in terms of constants and input column references. When `<expression>` refers to a unique source column, you do not need to include a table name qualifier. If `<expression>` is from another table or is not unique among the source columns, you need a table name qualifier. If `<expression>` is an empty string, the function looks up a zero-length varchar value in the lookup table. The function ignores trailing blanks in comparisons of `<expression >` and values in `<compare_column>`.

Note

You can specify more than one **<compare_column>** and **<expression>** pair—simply add additional pairs at the end of the function statement. The values must match for all specified pairs in order for the lookup function to find a matching row.

The lookup_seq function uses a value you provide (**<expression>**) to find a corresponding value in a different file or table (**<lookup_table>**). When multiple rows match, the function uses the row's sequence to determine the matching row.

More specifically, the function searches for the rows in the **<lookup_table>** where the value in the **<compare_column>** matches the value in **<expression>**. When the function finds multiple matching rows, it searches in the **<sequence_column>** for the row with the closest value less than or equal to the **<sequence_expression>**. If no row has a value less than or equal to the **<sequence_expression>**, the function finds the row with the closest value to the **<sequence_expression>**. For the matching row, the function returns the value in the **<result_column>**.

For example, if your source schema uses an employee ID to identify each row, and you want the employee's salary at the end of the previous year in your target schema, you can use the lookup_seq function to return the employee salary given the employee ID and the effective date of the salary. The salary returned will be the value corresponding to the latest effective date less than or equal to the value of **<sequence_expression>**.

In SQL terms, the lookup_seq function evaluates **<expression>** for each row, then determines which sequence column value meets the requirements:

```
SELECT MAX(<sequence_column>)
    FROM <lookup_table>, <source_table>
    WHERE <sequence_column> <= <sequence_expression>
        AND <compare_column> =
            <source_table.expression>)
```

Suppose this query stores the **<sequence_column>** value returned as **<sequence_result>**. Next, the function uses the **<sequence_result>** to find the proper **<result_column>**:

```
SELECT <result_column>
    FROM <lookup_table, source_table>
    WHERE <sequence_column> = <sequence_result>
        AND <compare_column> =
            <source_table.expression>
```

The value returned by these queries is the result of the lookup_seq function for the row.

You can specify multiple **<compare_column>** and **<expression>** pairs to uniquely identify the **<result_column>** value. However, the function wizard only provides fields for one pair; add extra **<compare_column>** and **<expression>** pairs to the output that the wizard generates.

Data Services always caches the comparison table when performing a lookup_seq function.

If the lookup_seq function does not find the value of **<expression>** in **<compare_column>**, then the function evaluates and returns the **<default_value>**.

Example

You can use the lookup_seq function to return a value from a slowly changing dimension table given an identifier. For example, suppose you have a source table that contains a numerical identifier, such as an employee number, and you want to retrieve the employee's salary at a specific time in the past. You can use the lookup_seq function to return the employee's salary on a particular date based on the employee number.

The source table contains the employee number and employee name.

You want the target table to contain the employee name and salary.

Use the lookup_seq function to translate the values from the source table to the desired values in the target table. The lookup_seq function uses a third, "translation," table.

To produce the desired target column, select the column in the target schema. Next, click the *Functions* button, located over the *Mapping* text box. In the function wizard, select *Miscellaneous_functions* under Function categories, then select *lookup_seq* under Function name. Enter the function parameters as follows:

Option	Value
Translate table	ODS_DS.SSB.FINANCE
Result column	Salary
Default value	'0'
Sequence column	EffectiveDate
Sequence Expression	'12.31.1999'
Compare column	EmployeeID
Expression	EmployID

The function wizard automatically produces the mapping text

```
lookup_seq(Ora_DS.RBH.FINANCE, SALARY, '0',
    EFFECTIVEDATE, '12.3.1.1999', Employee_ID, EmployID)
```

For each employee, this lookup_seq function returns the value from the salary column for that employee that is the most recent before December 31, 1999.

3.6.3.84 lower

Changes the characters in a string to lowercase.

Syntax

```
lower(<value,'locale'>)
```

Return value

varchar

The lowercase string. The return type is the same as **<value>**. Any characters that are not letters are left unchanged.

Where

<value>	The string to be modified.
<locale>	Optional parameter that converts the string to the specified locale. Note ISO 639 language code and ISO 3166 country code formats are supported.

Example

Function	Results
`lower('Accounting101')`	'accounting101'
`upper((LastName,1,1))|lower(substr(LastName,2,LENGTH(LastName)))`	The value in column `LastName` with the first letter uppercase and the rest of the value lowercase. Note that this example does not account for two-word last names.
`lower(LastName,'tr')`	The value in column `LastName` is converted to all lowercase. It is also converted to the Turkish locale, using the ISO 639 language code.

Related Information

ISO 639 language list

ISO 3166 Country Code list

3.6.3.85 lpad

Pads the string with characters in the left from a given pattern.

This function repeats the pattern at the beginning of the input string until the final string is the appropriate length. If the input_string is already longer than the expected length, then this function truncates the string.

Syntax

```
lpad(<input_string, size, pad_string>)
```

Return value

varchar

The modified string. The return type is the same as **<value>**. Any characters that are not letters are left unchanged.

Where

<input_string>	The string source.
<size>	An integer value indicating the number of characters in the return string.
<pad_string>	A character or set of characters that this function concatenate to **<input_string>**.

Example

Function	Results
lpad('Tanaka', 15, ' ')	`'         Tanaka'`
lpad(last_name, 25, ' ')	The value in the column last_name, padded with spaces to 25 characters on the left or truncated to 25 characters.

3.6.3.86 lpad_ext

Pads the left side of the string with logical characters from a given pattern.

Note

These logical characters prohibit this function from getting pushed down to the database.

This function repeats the pattern at the beginning of the input string until the final string is the appropriate length. If the input_string is already longer than the expected length, then this function truncates the string.

Syntax

```
lpad_ext(<input_string, size, pad_string>)
```

Return value

varchar

The modified string. The return type is the same as **<value>**. Any characters that are not letters are left unchanged.

Where

<input_string>	The string source.
<size>	An integer value indicating the number of characters in the return string.
<pad_string>	A logical character or set of logical characters that this function concatenates to the **<input_string>**.

Example

Function	Results
lpad_ext('Tanaka', 15, ' ')	`'        Tanaka'`
lpad_ext(last_name, 25, ' ')	The value in the column last_name, padded with spaces to 25 characters on the left or truncated to 25 characters.

lpad_ext and lpad functions exhibit the same behavior when the functions are evaluated within Data Services. In situations where the function is pushed down to the database, the database behavior may differ when 'input_string' and/or 'pad_string' parameters contain multibyte characters.

Function	Input	Output
lpad	("abc¶¶", 10,' ')	`'   abc¶¶'`
lpad_ext	("abc¶¶", 10,' ')	`'     abc¶¶'`
lpad	("abc¶¶", 10,'')	`'abc¶¶'`
lpad_ext	("abc¶¶", 10,'')	`'abc¶¶'`
lpad	("abcd", 10, '¶')	`'¶¶¶abcd'`
lpad_ext	("abcd", 10, '¶')	`'¶¶¶¶¶¶abcd'`

3.6.3.87 ltrim

Removes specified characters from the start of a string.

Syntax

```
ltrim(<input_string>, <trim_string>)
```

Return value

varchar

The modified string. The return type is the same as <input_string>.

Where

<input_string>	The string to be modified.
<trim_string>	The characters to remove from <input_string>.

The ltrim function is case-sensitive.

The function scans <input_string> left-to-right removing all characters that appear in <trim_string> until it reaches a character not in <trim_string>.

Example

Function	Results
`ltrim('Marilyn', ' ')`	`'Marilyn'`
`ltrim('ABCABCD', 'ABC')`	`'D'`
`ltrim('ABCABCD', 'EFG')`	`'ABCABCD'`
`ltrim('ABCDEABCDE', 'ABC'`	`'DEABCDE'`

To remove all leading blanks in a string, use ltrim as follows:

```
ltrim(EMPLOYEE.NAME, ' ')
```

where EMPLOYEE.NAME specifies the NAME column in the EMPLOYEE table.

3.6.3.88 ltrim_blanks

Removes blank characters from the start of a string.

Syntax

```
ltrim_blanks(<input_string>)
```

Return value

varchar

The modified string. The return type is the same as `<input_string>`.

Where

`<input_string>`	The string to be modified.

Example

Function	Results
`ltrim_blanks('    Marilyn')`	`'Marilyn'`
`ltrim_blanks(last_name)`	The value contained in the column last_name, with all leading blanks and control characters removed.

3.6.3.89 ltrim_blanks_ext

Removes blank and control characters from the start of a string.

Syntax

```
ltrim_blanks_ext(<input_string>)
```

Return value

varchar

The modified string. The return type is the same as **<input_string>**.

Where

<input_string>	The string to be modified.

Example

Function	Results
`ltrim_blanks_ext('    Marilyn')`	`'Marilyn'`
`ltrim_blanks_ext(last_name)`	The value contained in the column last_name, with all leading blanks removed.

3.6.3.90 mail_to

Captures the specified number of lines in the trace log and error log, packages the information as e-mail, and uses your Job Server computer's mail client to send e-mail messages to your local mail server for standard e-mail processing.

Syntax

```
mail_to(<recipients_list>, <subject>, <message>, <number_of_trace_lines>,
<number_of_error_lines>)
```

Return value

int

Returns 0 if function succeeds. Returns a non-zero integer if function fails.

Where

<recipients_list>	A string containing one or more recipient e-mail addresses separated by commas (,). This string cannot be empty and must contain valid, qualified e-mail address information.
<subject>	A string containing the subject of the e-mail. This string can be empty.
<message>	A string containing the e-mail message. This string can be empty.
<number_of_trace><_lines>	The number of lines from the end of the trace log file to append to the end of the e-mail. This input cannot be empty.
<number_of_error><_lines>	The number of lines from the end of the error log file to append at the end of the e-mail. This input cannot be empty.

Only use this function within a script.

To use this function, a mail client must be installed and running on the Job Server computer that calls the function. The login account for the mail client must have the same user name and password as the SAP Data Services service. The type of client varies by the operating system:

- If the Job Server is on a computer running the Windows operating system, then the mail client must comply with MAPI (message application programming interface). In addition, the mail client must be configured as the default mail client. For example, Microsoft Outlook is a MAPI-based mail client.
- If the Job Server is on a computer running the UNIX operating system, then the mail client must be mailx-compliant.

Note

If you do not have the required mail client, contact SAP Business User Support for assistance.

Example

Function	Results
`$myvar = mail_to('admin@company.com', 'Out of memory error in the SalesFact job. Please fix the error before running recovery job.',' ', 10, 10)`	The message is sent to one recipient.
`$myvar = mail_to('admin@company.com, manager@company.com', 'Out of memory error:' || systime(), 'Out of memory error while running the data flow:' || $dataflow_name || ' in the job:' || $job_name || '.', 10, 20)`	The message is sent to two recipients. The job name and data flow names are included in the text of the message as variables. Note that the software trims blank spaces from the end of strings; this example in-

Function	Results
	cludes a blank on the beginning of the next string. You can also concatenate a string with a single blank. In the script, type "$a = ;" where $a is the local integer variable defined in the work flow. Put the cursor just ahead of the semicolon before clicking the *Functions* button to construct a mail_to statement.

Note

Often, you list e-mail addresses as nicknames in your mail service address book. If your mail system is compatible with the software mail_to function, you can use these nicknames (comma separated) as values in the **`<recipients_list>`**. In this case, the software mailer program searches your e-mail address book for the nickname and uses the corresponding qualified e-mail address for message routing.

Limitation

The mail_to function might not work properly on Windows 2000 after downloading a security patch for Microsoft Outlook that includes email prompts. In this case, you have two options:

- To change the Outlook security settings to suppress prompts, see the instructions for Microsoft article 263297 on the following Web site: http://support.microsoft.com/default.aspx?scid=kb;en-us;Q263297&id=263297&SD=MSKB#OB.
  The article describes how to suppress prompting when sending e-mail from a particular computer. In summary, first create a security policy file on the Exchange Server (usually done by an administrator). In the security policy file, turn off prompting. Then add a registry key to the Job Server (client) computer. When the Job Server computer tries to send an e-mail, the Outlook client first checks the registry key; if the key is set, Outlook checks the security policy file on the Exchange Server and suppresses prompts.
- Use the smtp_to function instead.

Related Information

Microsoft Help and Support

3.6.3.91 match_pattern

Allows you to match a whole input string to simple patterns that Data Services supports for this function. This function does not match substrings.

Return Value

integer	Returns 1 for a match, otherwise 0.

Syntax

```
match_pattern(<input_string>,<pattern_string>)
```

Where

input_string	String to be matched. Supports UNICODE characters.
pattern_string	Pattern you want to find in a whole input string. Substring matches are not supported.

Use the following characters to create a pattern:

X	Represents uppercase characters; general category Lu as per Unicode 4.0 specification (for example, Latin, Greek, Cyrillic, Armenian, Deseret, and archaic Georgian).
x	Represents non-uppercase characters: • Ll—Lowercase letter (for example Latin, Greek, Cyrillic, Armenian, Deseret, and archaic Georgian). • Lt—Title letters (for example Latin capital letter D with small letter Z). • Lm— Modifier letter (for example acute accent, grave accent). • Lo—Other letters (including Chinese, Japanese, and so on).
9	Represents numbers.
\	Escape character.
*	Any characters occurring zero or more times.
?	Any single character occurring once and only once.
[ ]	Any one character inside the braces occurring once.
[!]	Any character except those after the exclamation point (i.e. [!12] can allow any, say zip code, that does not start with a 1 or 2.

All other characters represent themselves. If you want to specify a special character as itself, then it has to be escaped. For example, [!9] means except any digit. To specify except nine, the correct pattern is [!\9].

The following table displays pattern strings that represent example values:

Example Value	Pattern string
Henrick	Xxxxxxx
DAVID	XXXXX
Tom Le	Xxx Xx
Real-time	Xxxx-xxxx
JJD)$@&*hhN8922hJ7#	XXX)$@&*xxX9999xX9#
1,553	9,999
0.32	9.99
-43.88	-99.99
Returns names with last name Jones	*Jones
Returns Henrick1 or HenrickZ	Henrick?
Returns David1 or David2 or David3	David[123]

Example

Use the `match_pattern` function in the Validation transform or in a WHERE clause of Query. The input string can be from sources such as columns, variables, or constant strings.

Use Case	Pattern	Function Call	Results

To match a zip code except one that begins with 1 or 2.	'[!12]9999'	`if (match_pattern('15014', '[!12]9999') <> 0)`
		`   print('matched');`
		`else`
		`   print('not matched');`
			Function prints "not matched".

To match a zip code except one that begins with 1 or 2.	'[!12]9999'	`if (match_pattern('55014', '[!12]9999') <> 0)`
		`   print('matched');`
		`else`
		`   print('not matched');`
			Function prints "matched".

To process only customer phone numbers that fit the same pattern.	'999-999-9999'	`WHERE MATCH_PATTERN(CUSTOMER.PHONE_NUM,'999-999-9999') <> 0`	Phone numbers that do not match the pattern throw error 0.
To check a string against a com-	'XXX)$@&*xxX9999xX9#'	`if(match_pattern('JJD)$@&*hhN8922hJ7#', 'XXX)$@&*xxX9999xX9#') <> 0)`	The result for this call is "matched".

Use Case	Pattern	Function Call	Results
plex pattern and print result to trace log.		<code>print ('matched');</code> <code>else</code> <code>print('not matched');</code>	

Related Information

literal [page 1596]

3.6.3.92 match_regex

Matches whole input strings to the pattern that you specify with regular expressions (regular expressions based on the POSIX standard) and flags. POSIX refers to the POSIX.1 standard (IEEE Std 1003.1) which defines system interfaces and headers with relevance for string handling and internationalization. The XPG3, XPG4, Single Unix Specification (SUS) and other standards include POSIX.1 as a subset. The patterns listed here in the *Reference Guide* adhere to the current standard. See *http://icu.sourceforge.net/userguide/regexp.html* for more information and updates. This function does not match substrings.

Syntax

```
match_regex (input_string, regular_expression_pattern, flags)
```

Return Value

`integer`	Returns `1` for a match, otherwise `0`.

Where

`input_string`	String to be matched. Supports UNICODE characters.
`regular_expression_pattern`	Pattern you want to find in a whole input string. Substring matches are not supported. Provide the pattern in regular expression format with a varchar data type.

`flags`	Allows you to specify additional behavior that you want to occur while Data Services searches the input_string for pattern matches. Enter NULL if you do not want to specify a flag. You can combine the options for flags using a comma. Flag options are case sensitive and need to be specified in upper case.

You can use the following regular expression patterns in the pattern parameter:

Character	Description
\a	Match a BELL, \u0007.
\A	Match at the beginning of the input. Differs from ^ in that \A will not match after a new line within the input.
\b, outside of a [Set]	Match if the current position is a word boundary. Boundaries occur at the transitions between word (\w) and non-word (\W) characters, with combining marks ignored. For better word boundaries, see ICU Boundary Analysis.
\b, within a [Set]	Match a BACKSPACE, \u0008.
\B	Match if the current position is not a word boundary.
\cX	Match a control-X character.
\d	Match any character with the Unicode General Category of Nd (Number, Decimal Digit).
\D	Match any character that is not a decimal digit.
\e	Match an ESCAPE, \u001B.
\E	Terminates a \Q ... \E quoted sequence.
\f	Match a FORM FEED, \u000C.
\G	Match if the current position is at the end of the previous match.
\n	Match a LINE FEED, \u000A.
\N{UNICODE CHARACTER NAME}	Match the named character.
\p{UNICODE PROPERTY NAME}	Match any character with the specified Unicode Property.
\P{UNICODE PROPERTY NAME}	Match any character not having the specified Unicode Property.
\Q	Quotes all following characters until \E.
\r	Match a CARRIAGE RETURN, \u000D.

Character	Description
\s	Match a white space character. White space is defined as [\t\n\f\r\p{Z}].
\S	Match a non-white space character.
\t	Match a HORIZONTAL TABULATION, \u0009.
\uhhhh	Match the character with the hex value hhhh.
\Uhhhhhhhh	Match the character with the hex value hhhhhhhh. Exactly eight hex digits must be provided, even though the largest Unicode code point is \U0010ffff.
\w	Match a word character. Word characters are [\p{Ll}\p{Lu}\p{Lt}\p{Lo}\p{Nd}].
\W	Match a non-word character.
\x{hhhh}	Match the character with hex value hhhh. From one to six hex digits may be supplied.
\xhh	Match the character with two digit hex value hh.
\X	Match a Grapheme Cluster.
\Z	Match if the current position is at the end of input, but before the final line terminator, if one exists.
\z	Match if the current position is at the end of input.
\n	Back Reference. Match whatever the nth capturing group matched. n must be a number > 1 and < total number of capture groups in the pattern. Note: Octal escapes, such as \012, are not supported in ICU regular expressions.
[pattern]	Match any one character from the set. See UnicodeSet for a full description of what may appear in the pattern.
.	Match any character.
^	Match at the beginning of a line.
$	Match at the end of a line.
\	Quotes the following character. Characters that must be quoted to be treated as literals are * ? + [( ) { } ^ $ | \ . /

You can use the following regular expression operators in a pattern parameter:

Operator	Description
|	Alternation. A|B matches either A or B.
*	Match 0 or more times. Match as many times as possible.

Operator	Description
+	Match 1 or more times. Match as many times as possible.
?	Match zero or one times. Prefer one.
{n}	Match exactly n times.
{n,}	Match at least n times. Match as many times as possible.
{n,m}	Match between n and m times. Match as many times as possible, but not more than m.
*?	Match 0 or more times. Match as few times as possible.
+?	Match 1 or more times. Match as few times as possible.
??	Match zero or one times. Prefer zero.
{n}?	Match exactly n times.
{n,}?	Match at least n times, but no more than required for an overall pattern match.
{n,m}?	Match between n and m times. Match as few times as possible, but not less than n.
*+	Match 0 or more times. Match as many times as possible when first encountered, do not retry with fewer even if overall match fails. Possessive match.
++	Match 1 or more times. Possessive match.
?+	Match zero or one times. Possessive match.
{n}+	Match exactly n times.
{n,}+	Match at least n times. Possessive match.
{n,m}+	Match between n and m times. Possessive match.
(...)	Capturing parentheses. Range of input that matched the parenthesized subexpression is available after the match.
(?: ...)	Non-capturing parentheses. Groups the included pattern, but does not provide capturing of matching text. Somewhat more efficient than capturing parentheses.
(?> ...)	Atomic-match parentheses. First match of the parenthesized subexpression is the only one tried; if it does not lead to an overall pattern match, back up the search for a match to a position before the "(?>"
(?# ...)	Free-format comment (?# comment).
(?= ...)	Look-ahead assertion. True if the parenthesized pattern matches at the current input position, but does not advance the input position.

Operator	Description
(?! ...)	Negative look-ahead assertion. True if the parenthesized pattern does not match at the current input position. Does not advance the input position.
(?<= ...)	Look-behind assertion. True if the parenthesized pattern matches text preceding the current input position, with the last character of the match being the input character just before the current position. Does not alter the input position. The length of possible strings matched by the look-behind pattern must not be unbounded (no * or + operators).
(?<! ...)	Negative look-behind assertion. True if the parenthesized pattern does not match text preceding the current input position, with the last character of the match being the input character just before the current position. Does not alter the input position. The length of possible strings matched by the look-behind pattern must not be unbounded (no * or + operators).
(?ismx-ismx: ...)	Flag settings. Evaluate the parenthesized expression with the specified flags enabled or disabled.
(?ismx-ismx)	Flag settings. Change the flag settings. Changes apply to the portion of the pattern following the setting. For example, (?i) changes to a case insensitive match.

You can use the following flags in the flag parameter:

Flag Options	Description
'CASE_INSENSITIVE'	If set, matching will take place in a case-insensitive manner.
'COMMENTS'	If set, allows use of white space and #comments within patterns.
'DOTALL'	If set, a "." in a pattern will match a line terminator in the input text. By default, it will not. Note that a carriage-return / line-feed pair in text behave as a single line terminator and match a single "." in a regular expression pattern.

Example

Use the Match_regex function in the Validation transform by accessing the Smart Editor or Function wizard or in a WHERE clause of a Query. The input string can be from sources such as columns, variables, or constant strings.

Use Case	Pattern	Function Call
To match phone numbers in (408)-933-6000 format.	'([0-9]{3}-[0-9]{3}-[0-9]{4}'	`match_regex (pho_number,'([0-9]{3}-[0-9]{3}-[0-9]{4}',NULL)`

Use Case	Pattern	Function Call
To match a string that starts with "topicA" regardless of case.	'topicA.*'	`match_regex (subject, 'topicA.*','CASE_INSENSITIVE')`

Related Information

literal [page 1596]

3.6.3.93 match_simple

Allows you to match a whole input string to simple patterns that Data Services supports for this function. This function does not match substrings.

Return Value

integer	Returns `1` for a match, otherwise `0`.

Syntax

```
match_simple(<input_string>,<pattern_string>)
```

Where

`<input_string>`	String to be matched. Supports UNICODE characters.
`<pattern_string>`	Pattern you want to find in a whole input string.

Use the following characters to create a pattern:

.	Represents any single character.
*	Represents any character zero or more times.
#	Represents any single alphabetic character including non-English letters.

$	Represents any alphabetic character, including non-English letters, zero or more times.
+	Matches the previous character one or more times.
(string)+	Matches the string one or more times.
[number1..number2]	Numeric range (integers only). Matches any number between number1 and number2.
\	Escape character
;	OR operator. If the data matches any of the identified patterns, the result is TRUE. Enclose the list with curly brackets {}. Example: `{ABC+;XYZ*}` If the data matches either `ABC+` or `XYZ*`, the result is TRUE.
<>	NOT operator. Specify the pattern after the <>. Example: `<><pattern>`
{EMPTY} and {empty}	Special predefined patterns that match empty data.
{NULL} and {null}	Special predefined patterns that match NULL data.

If the pattern is empty, then it matches all data.

If the value of a pattern column is NULL, then it will not match with any value.

All other characters represent themselves. If you want to specify a special character as itself, then it has to be escaped.

The following table displays pattern strings that represent example values:

Example Value	Pattern string
ACCT1234567	ACCT*
ZIP10000 to ZIP99999	ZIP[10000..99999]
ACCT123 or ACCOUNT234	{ACCT*;ACCOUNT*}
www.anything.com	www.$.com

Example

Use the match_simple function in the Validation transform or in a WHERE clause of Query. The input string can be from sources such as columns, variables, or constant strings. The following example illustrates sample code used in a script.

Use Case	Pattern	Function Call	Results

To match account numbers from ACCT1 to ACCT5000	ACCT[1..5000]	`if (match_simple('ACCT14',`
		`'ACCT[1..5000]')) <> 0`
		`print ('matched');`
			Function prints "matched".

Use Case	Pattern	Function Call	Results

		`else`
		`print('not matched');`	

3.6.3.94 max

Returns the maximum value from a list.

Syntax

```
max(<value_list>)
```

Return value

any type

The maximum value of the column values. The return type is the same as the values in **`<value_list>`**.

Where

`<value_list>`	The source values for which to identify a maximum.

Example

To calculate the maximum value in the salary column of a table, use the max function in a query:

- In the *Mapping* tab of the query editor, enter:

  ```
  max(SALARY)
  ```

- In the *Group By* tab in the query editor, specify the columns for which you want to find the maximum salary, such as the department column. For each unique set of values in the group by list, such as each unique department, Data Services calculates the maximum salary.

3.6.3.95 min

Returns the minimum value from a list.

Syntax

```
min(<value_list>)
```

Return value

any type

The minimum value of the column values. The return type is the same as the values in `<value_list>`.

Where

`<value_list>`	The source values for which to identify a minimum.

Example

To calculate the minimum value in the salary column of a table, use the min function in a query:

- In the *Mapping* tab of the query editor, enter:

  ```
  min(SALARY)
  ```

- In the *Group By* tab in the query editor, specify the columns for which you want to find the minimum salary, such as the department column. For each unique set of values in the group by list, such as each unique department, Data Services calculates the minimum salary.

3.6.3.96 mod

Returns the remainder when one number is divided by another.

Syntax

```
mod(number1, number2)
```

Return Value

Depends on the input numbers.

Where

`<number1>`	Number to be divided.
`<number2>`	Divisor of first number.

Details

Returns the remainder when one number is divided by another.

Note that the % operator used in Data Services syntax produces the same result.

Example

Function	Result
`mod(100.10,10));`	`0.100000`

3.6.3.97 month

Determines the month in which the given date falls.

Syntax

```
month(<date1>)
```

Return value

int

The number from 1 to 12 that represents the month component of `<date1>`.

Where

`<date1>`	The source date.

Example

Function	Results
`month(to_date('Jan 22, 1997', 'mon dd, yyyy'))`	1
`month(to_date('3/97', 'mm/yy'))`	3

3.6.3.98 num_to_interval

Converts a numeric value to an interval.

Syntax

```
num_to_interval(<number1, format>)
```

Return value

interval

The converted interval.

Where

`<number1>`	The value of type `int`, `real`, `decimal`, or `numeric` to convert.
`<format>`	A string describing the format of the interval. Choose from the following values: D: Days H: Hours M: Minutes S: Seconds

Example

Function	Results
`num_to_interval(elapsed_days, 'D')`	The value from the column `elapsed_days` converted to an interval of days.
`start_time + num_to_interval(elapsed_seconds, 'S')`	This example assumes that it is acting on an input schema which contains (at least) the columns 'start_time' and 'elapsed_seconds' (for example, the start_time might be '2005-12-01 00:00:00' and elapsed_seconds might be 200). So, this example indicates a time which is the number of elapsed seconds after the start time ('2005-12-01 00:03:20').

3.6.3.99 nvl

Replaces NULL values.

Syntax

```
nvl(<expression1>, <replacement_value>)
```

Return value

any type

The value of **<expression1>** if not NULL, otherwise, the value of **<replacement_value>**.

Where

<expression1>	The value to be tested for NULL.
<replacement_value>	The value to replace **<expression1>** if **<expression1>** is NULL. **<replacement_value>** must be the same data type as **<expression1>**.

Example

Function	Results
`nvl(modification_date, sysdate())`	If the column `modification_date` for a row hasn't been set, this function inserts today's date.
`nvl(lookup(r3..vbpa, kunnr, 'NULL', vbeln, vbak.vbeln, posnr, vbap.posnr, parvw, 'RE'), lookup(r3..vbpa, kunnr, 'NULL', vbeln, vbak.vbeln, posnr, vbap.posnr, parvw, 'RG'))`	Both expressions are determined by the result of lookup functions.

3.6.3.100 power

Returns the value of the given expression to the specified power.

Syntax

```
power(num, num)
```

Return Value

Float data type

Where

<num>	Numeric expression representing base number.
<num>	Numeric expression representing power.

Details

Returns the value of the given expression to the specified power.

Example

Function	Results
`power(2.2,3));`	`10.648000`

3.6.3.101 previous_row_value

Returns the column value of the previous row.

Note

Each call to the previous_row_value() function will return the value stored during the previous call of this function. In other words, if the function is not called for each row, the results of this function might not be what you expect (not the previous row's value). This scenario can happen if you, for example, use the previous_row_value() inside an ifthenelse() function:

```
If_then_else (table1.status = 'new', 0 , previous_row_value(table1.value))
```

A better solution in the example above would be :

```
If_then_else (table1.status = 'new', 0 , 1) * previous_row_value(table1.value)
```

or use two different queries: one for the previous_row_value() and one for the final result including the if_then_else().

Syntax

```
previous_row_value(expression)
```

Return Value

Data type of the input parameter. First row always returns `NULL`.

Where

`<expression>`	Valid Input expression.

Details

This function is useful in Query transforms. It returns the previous row's value. For example, the input stream of the column might be `1;2;3;4` for the first four rows. The function returns `NULL;1;2;3`.

Example

Following is list of records of sales figures for each day:

Date Revenue

rec 1 1/1/2005 1000

rec 2 1/2/2005 1100

rec 3 1/3/2005 900

rec 4 1/4/2005 1200

The requirement is to calculate the delta of the revenue with the previous day. So the query uses an order by on Date and calculate Revenue - Previous_Row_Value (revenue) which results in:

Date Revenue Delta = Revenue - Previous_Row_Value

rec 1 1/1/2005 1000 NULL

rec 2 1/2/2005 1100 +100

rec 3 1/3/2005 900 -200

rec 4 1/4/2005 1200 +300

3.6.3.102 print

Prints the given string to the trace log.

Syntax

```
print(<input_string>)
```

Return value

int

Value is **<input_string>** when the string contains valid data. Value is NULL and no string prints when the string contains NULL data.

Where

`<input_string>`	The message to be written to the trace log.

Example

Function	Results
`print('Reached decision point for running full or incremental data flows')`	Writes "Reached decision point for running full or incremental flows" to trace log and returns `<input_string>`.
`print('The date is: [$start_date]')`	Writes "The date is 2000.06.03" to trace log and returns `<input_string>`.
`print('[$month_sal*12]')`	Writes "48000" to trace log and returns `<input_string>`.
`print('Total Sal is: [$month_sal*12]');`	Writes "Total Sal is: 48000" to trace log and returns `<input_string>`.
`print('The return value from the SQL() function is > [$y]');`	Writes "The return value from the SQL() function is > 23456" to trace log and returns `<input_string>`.

3.6.3.103 pushdown_sql

Allows you to create dynamic WHERE clauses.

Syntax

```
pushdown_sql (<datastore>, <input_string>)
```

Return Value

None.

Where

`<datastore>`	The name of the datastore containing the data you want to retrieve. Data Services creates a WHERE clause and pushes it down to this database. Surround the datastore name by single quotes.
`<input_string>`	A character string that forms the WHERE clause. Surround the character string by single quotes. Typically, this is a column from another source to the query, such as an XML message. Delimit columns or parameters with curly braces. For example, {XML_IN.STATUS_QUERY}. If the string contains a curly brace, use the backslash escape key to delimit the curly brace.

Details

The pushdown_sql function allows you to create WHERE clauses that change based on data input. With the pushdown_sql function, the WHERE clause need not be pre-specified. The pushdown_sql function is particularly useful in real-time jobs, if you want to select data based on input from an XML message.

Unlike other functions, the pushdown_sql function can only be used in the WHERE clause of a Query transform. You cannot use the pushdown_sql function in other places, such as in a query's mapping, in a conditional, or in a script.

Data Services must be able to push the WHERE clause that it creates from this function to the database. This function works best, therefore, when used in a Query transform where the immediate input is the table source where you want to push the WHERE clause.

Data Services does not parse the SQL contained in the input string. Therefore, the input must be well-formed with correct syntax.

Note

Data Services does not allow use of the backslash escape key to delimit curly braces within the pushdown_sql function. So, if your input string contains a curly brace, you must make the string into a variable. Therefore, instead of entering 'a\{b\}c', you would pass your data through as 'a{$x}c' where $x = '\{b\}'.

Example

Suppose the datastore EC_DS contains the table BIKES, which stores information about different models. And suppose the QUERY_REQUEST column in the XML_IN message contains requests for information from this table. For example, a value in the QUERY_REQUEST column might be:

```
TYPE = 'MOUNTAIN' and PRICE < 1500
```

In a data flow used in a real-time job, you can use the pushdown_sql function in a query to select data from the BIKES table based on the data in the XML_IN message. You can return the data in another XML message.

Function	Results

`pushdown_sql ('EC_DS',`
`'{XML_IN.QUERY_REQUEST}')`
	Data Services includes the following WHERE clause in the SQL SELECT statement:
	`WHERE TYPE = 'MOUNTAIN' and PRICE <`
	`1500`

3.6.3.104 quarter

Determines the quarter in which the given date falls.

Syntax

```
quarter(<date1>)
```

Return value

int

The number from 1 to 4 that represents the quarter component of **<date1>**.

Where

<date1>	The source date.

Example

Function	Results

`quarter(to_date('Jan 22, 1997', 'mon`
`dd, yyyy'))`
	`1`

`quarter(to_date('5/97', 'mm/yy'))`	`2`

3.6.3.105 raise_exception

Calling this function causes an exception to be generated.

Syntax

```
raise_exception(<error_msg>)
```

Return Value

int

Returns '1' always.

Where

`<error_msg>`	The string which will be written to the Job Server's error log.

Details

The work flow or job may or may not be terminated based on whether a try-catch block surrounds the call.

Example

```
ifthenelse(sal < 1000000, 0, raise_exception('Salary exceeds 1 million dollars.'))
```

3.6.3.106 raise_exception_ext

Calling this function causes an exception to be generated with an exit code.

Syntax

```
raise_exception_ext(<error_msg>),(<exit_code>)
```

Return Value

int

Returns '1' always.

Where

<error_msg>	The string which will be written to the Job Server's error log.
<exit_code>	Code the job exits with, if the exception is not caught in a try/catch block. Use a number in range 1 to 255 (zero means "success" to all operating systems).

Details

The work flow or job may or may not be terminated based on whether a try-catch block surrounds the call.

Example

```
ifthenelse(sal < 1000000, 0, raise_exception_ext('Salary exceeds 1 million
dollars.', sal/1000000 + 1))
```

3.6.3.107 rand

Returns a random number between 0 and 1.

Syntax

```
rand()
```

Return value

real

The random number. The return value is between 0 and 1.

Example

Function	Results
`100 * rand()`	A random number between 0 and 100.

3.6.3.108 rand_ext

Similar to, and more powerful than the rand function, rand_ext returns a random number between 0 inclusive and 1 exclusive. This function uses the linear-congruential generator (LCG) algorithm, $x_n = (ax_{n-1} + b)\mathrm{mod}\ m$ where:

x_n is an integer from 0 to m-1 and the initial value of x_n is called the "seed" (x_0). For each call to the random number generator, Data Services calculates a new x_n by taking the value of the previous result x_{n-1}, multiplying by *a*, adding *b*, then taking the remainder mod *m*.

Data Services uses this formula to generate an integer from 0 to m-1. After Data Services calculates x_n, it divides that number by *m* to obtain a number equal to or greater than 0 and less than 1.

By specifing the same seed number, you can regenerate an exact number sequence (for use in repeat experiments).

Syntax

```
real rand_ext([seed])
```

Return value

real

The random number. The return value is between 0 and 1.

Where

<seed>	(Optional) Can be any positive integer greater than or equal to 0. If unspecified, Data Services uses the current time to create a seed.

Example

Function	Results
`100 * rand_ext()`	A random number between 0 and 100.

3.6.3.109 replace_substr

Takes an input string, replaces each occurrence of a specified substring with a specified replacement, and returns the result.

Syntax

```
replace_substr(<in_str, search_str, replace_str>)
```

Return Value

varchar

Where

<in_str>	The input string that you are changing. If NULL, return will be NULL.
<search_str>	String to search for. If **<search_string>** is NULL, varchar is returned and will be the same as in_str.
<replace_str>	If **<replace_string>** is omitted or NULL, all occurrences of **<search_string>** are removed.

Example

Function	Result

`replace_substr('a penny saved is a`
`penny earned',`
`'penny', 'million')'`
	`'a million saved is a million earned'`

3.6.3.110 replace_substr_ext

Takes an input string, replaces specified occurrences of a specified sub-string with a specified replacement and returns the result. It is also possible to search for the following:

- a hexadecimal value that refers to a UNICODE character
- a non-printable character reference such as a form feed or new line

Syntax

```
replace_substr_ext(in_str, search_str,
replace_str,start_at_occurance,number_of_occurances)
```

Return Value

varchar

Where

`in_str`	The input string that you are changing. If NULL, returns NULL.
`search_str`	String to search for: You can use /x0000 to specify the hexadecimal value for a special character. For example, if you use /x000A, then if Data Services encounters /x it will convert the next 4 characters to a hexadecimal value. This function converts the hexadecimal value to a UNICODE character. This option provides more flexibility when you use a search string. You can also represent special characters using the escape character '/'. The following are supported. `/a` Bell (alert) `/b` Backspace `/f` Formfeed `/n` New line `/r` Carriage return `/t` Horizontal tab `/v` Vertical tab To include the escape character '/' in the search string, escape it using '//'. For example, if the input is 'abc/de', Data Services converts `search_str` to 'abcde'. While if the input is 'abc//de', Data Services converts `search_str` to 'abc/de'. If `search_str` is NULL, Data Services returns a varchar with the data in `in_str`.
`replace_str`	String that replaces `search_str`. If **`<replace_string>`** is omitted or NULL, all occurrences of **`<search_str>`** are removed.

`start_at_occurence`	Start replacing at this occurrence. If NULL, start at the 1st occurrence. For example, enter 2 to replace or remove the second occurrence of a `search_str`.
`number_of_occurences`	Number of occurrences to replace. If NULL, replace all occurrences. For example, enter 2 to replace or remove two sequential occurrences of the `search_str`.

Example

Function	Result
Replace 'a' with 'B' starting from second occurrence and replaces two occurrences: `replace_substr_ext('ayyyayyyayyyayyy', 'a', 'B', 2, 2)'`	`'ayyyByyyByyyayyy'`
Search a string containing 'a' followed by a new line and replace it with 'B' starting from second occurrence and replaces two occurrences: `replace_substr_ext('ayyya</n>yyya</n>yyyayyy', 'a/n', 'B', 2, 2)`	`'ayyyByyyByyyayyy'`
Search a string containing 'a' followed by a new line and replace it with 'B' starting from second occurrence and replaces two occurrences: `replace_substr_ext('ayyya</n>yyya</n>yyyayyy', 'a/x000a', 'B', 2, 2)`	`'ayyyByyyByyyayyy'`

3.6.3.111 repository_name

Returns a database connection string and owner name. For example: `beq-local.DBUser.` This is the ID for the repository from which the job is run.

Syntax

```
repository_name()
```

Return Value

varchar

Example

```
print('Repository Name: [repository_name()]')
```

3.6.3.112 round

Rounds a given number to the specified precision.

Syntax

```
round(<num1>, <precision>)
```

Return value

decimal, double, int, or real

The rounded number. The return type is the same as the original number, **<num1>**.

Where

<num1>	The source number.
<precision>	An integer indicating the number of decimals in the result. If **<precision>** is negative, digits left of the decimal point are rounded.

Example

Function	Results
`round(120.12345, 2)`	120.12
`round(120.12999, 2)`	120.13
`round(120, -2)`	100
`round(120.123, 5)`	120.12300

3.6.3.113 rpad

Pads a string with characters from a given pattern.

The function repeats the pattern at the end of the input string until the final string is the appropriate length. If the input string is already longer than the expected length, this function truncates the string.

Syntax

```
rpad(<input_string>, <size>, <pad_string>)
```

Return value

varchar

The new string.

Where

`<input_string>`	The source string.
`<size>`	An integer value indicating the number of characters in the resulting string.
`<pad_string>`	A character or set of characters that this function concatenates to `<input_string>`.

Example

Function	Results
`rpad('Tanaka',15,' ')`	`'Tanaka         '`
`rpad(last_name,25,' ')`	The value in the column `last_name`, padded with spaces to 25 characters, or truncated to 25 characters.

3.6.3.114 rpad_ext

Pads a string with logical characters from a given pattern.

Note

These logical characters prohibit this function from getting pushed down to an Oracle database.

The function repeats the pattern at the end of the input string until the final string is the appropriate length. If the input string is already longer than the expected length, this function truncates the string.

Syntax

```
rpad_ext(<input_string>, <size>, <pad_string>)
```

Return value

varchar

The new string.

Where

`<input_string>`	The source string.
`<size>`	An integer value indicating the number of characters in the resulting string.
`<pad_string>`	A character or set of characters that this function concatenates to **`<input_string>`**.

Example

Function	Results
`rpad_ext('Tanaka',15,' ')`	`'Tanaka         '`
`rpad_ext(last_name,25,' ')`	The value in the column `last_name`, padded with spaces to 25 characters, or truncated to 25 characters.

rpad_ext and rpad functions exhibit the same behavior when the functions are evaluated within Data Services. In situations where the function is pushed down to the database, the database behavior may differ when 'input_string' and/or 'pad_string' parameters contain multibyte characters.

Function	Input	Output
rpad	("abc¶¶", 10,' ')	`'abc¶¶   '`
rpad_ext	("abc¶¶", 10,' ')	`'abc¶¶     '`
rpad	("abc¶¶", 10,'')	`'abc¶¶'`
rpad_ext	("abc¶¶", 10,'')	`'abc¶¶'`
rpad	("abcd", 10, '¶')	`'abcd¶¶¶'`
rpad_ext	("abcd", 10, '¶')	`'abcd¶¶¶¶¶¶'`

3.6.3.115 rtrim

Removes specified characters from the end of a string.

Syntax

```
rtrim(<input_string>, <trim_string>)
```

Return value

varchar

The modified string. The return type is the same as `<input_string>`.

Where

`<input_string>`	The string to be modified.
`<trim_string>`	The characters to remove from `<input_string>`.

The function scans `<input_string>` right-to-left removing all characters that appear in `<trim_string>` until it reaches a character not in `<trim_string>`.

Removes trailing blanks only if `<trim_string >`contains trailing blanks. If the length of the modified string becomes zero after trimming, the function returns '' (empty string).

To remove all trailing blanks in a string, use the rtrim_blanks function.

Example

Function	Results
`rtrim('Marilyn ', ' ')`	'Marilyn'
`rtrim('ZABCABC', 'ABC')`	'Z'
`rtrim('ZABCABC', 'EFG')`	'ZABCABC'

3.6.3.116 rtrim_blanks

Removes blank characters from the end of a string.

Syntax

```
rtrim_blanks(<input_string>)
```

Return value

varchar

The modified string. The return type is the same as **<input_string>**.

Where

<input_string>	The string to be modified.

If the length of the modified string becomes zero after trimming, the function returns '' (empty string).

Example

Function	Results
`rtrim_blanks('Marilyn ')`	'Marilyn'
`rtrim_blanks(last_name)`	The value contained in the column `last_name` with trailing blanks removed.

3.6.3.117 rtrim_blanks_ext

Removes blank and control characters from the end of a string.

Syntax

```
rtrim_blanks_ext(<input_string>)
```

Return value

varchar

The modified string. The return type is the same as **<input_string>**.

Where

<input_string>	The string to be modified.

If the length of the modified string becomes zero after trimming, the function returns '' (empty string).

Example

Function	Results
`rtrim_blanks('Marilyn ')`	'Marilyn'
`rtrim_blanks(last_name)`	The value contained in the column `last_name` with trailing blanks and control characters removed.

3.6.3.118 sap_openhub_processchain_execute

Performs the following tasks:

- Starts the process chain that extracts data from an InfoProvider (InfoArea, InfoCube, or DataStore object) on SAP NetWeaver Business Warehouse and loads the extracted data into an Open Hub Destination table.
- Monitors the process chain status and the Open Hub Destination request notification.

When the function returns successfully, an Open Hub table source in SAP Data Services can then read the data from the Open Hub Destination table.

Note

You can only use this function in a script. It is not valid in a query or audit object.

Below is the function syntax as a reference. The function wizard is explained in the next section.

Syntax

```
sap_openhub_processchain_execute('<datastore>','<open_hub_table>',
'<process_chain>',<$logid>,<$ReturnTxt>)
```

Where

<datastore>	Specifies the datastore name. You can specify either a constant string or a substitution parameter. The data type is varchar(256).

	Note The maximum length depends on the Data Services repository type. For most repository types the maximum length is 256, for MySQL the length is 64, and for MS SQL server the length is 128.
`<open_hub_table>`	Specifies the Open Hub Destination table. You can specify either a constant string or a substitution parameter. The data type is varchar(30).
`<process_chain>`	Specifies the name of the process chain that extracts data from the InfoProvider in SAP NetWeaver BW and loads the data to the Open Hub Destination table. You can specify either a constant string or a substitution parameter. The data type is varchar(30).
`<$logid >`	(Optional) Specifies a variable to obtain a value that depends on the function return value (see "Relationship between return value and value of logid variable" below). The required variable data type is varchar(25).
`<$ReturnTxt>`	(Optional) Specifies a variable to retrieve the description of the return status of the process chain. The required variable data type is varchar, and you can define the length you want for this variable.

Return value

varchar(1)

Returns one of the following values.

Return value	Description
B	Open Hub Destination is being read by another user.
E	Data Services error while executing the function.
R	Process chain execution failed with errors in BW system.
X	Process chain execution has been canceled in BW system.
S	Function successfully executed the Open Hub extraction Data Transfer Process (DTP) and received extraction request notification.

Relationship between return value and value of logid variable

The value of the logid output variable depends on the function return value, as the following table shows.

Return value	$logid variable value	$ReturnText variable value	Action
B	Process chain log ID of the other user that is currently reading the Open Hub Destination	Status of current Open Hub extraction	Either wait and try again or stop executing the data flow that contains the Open Hub Destination table.
E	Data Services error log number	Data Services error text	Stop executing the data flow that contains the Open Hub Destination table, and analyze the Data Services error.
R	Your process chain log ID	Error from process chain	Stop executing the data flow that contains the Open Hub Destination table, and use the log ID in the BW system to see the detail state of the process chain error.
X	Your process chain log ID	Error from process chain	Stop executing the data flow that contains the Open Hub Destination table, and use the log ID in the BW system to see the detail state of the process chain error.
S	Open Hub extraction request ID	Status of your Open Hub extraction	Use the request ID in the BW system to obtain detail loading statistics (such as number of packets loaded and number of records loaded).

Example

The following sample script commands check the return value, generate an exception, and print the error if the function is not successful.

```
$status = sap_openhub_processchain_execute('open_hub_datastore',
'Materials','Materials_PC',<$lpcogid>,
<$returntxt>);
If ($status != 'S') raise_exception ('Materials_PC process chain execution failed
' || $returntxt);
```

Restrictions

The following are restrictions for using Open Hub Destinations:

- Only one job at a time can read an Open Hub Destination table.
- A process chain of an Open Hub Destination can contain only one of its Data Transfer Processes (DTPs).
- A process chain cannot contain DTPs for more than one Open Hub Destination.

3.6.3.118.1 To define an sap_openhub_processchain_execute function

1. To access the function wizard for sap_openhub_processchain_execute from the Script Editor, click *Functions* or *...* at the top of the window.
2. Select *sap_openhub_processchain_execute* from the list of functions.
   The *Define Parameters* window opens.
3. Select an SAP BW Source datastore name from the drop-down list. You can also select a substitution variable from the list. If you type in a datastore name, put single quotes around the name.
4. Select the name of a Open Hub table from the drop-down list. Only the names of the imported Open Hub Tables appear in this list.
5. Select the name of a Process Chain from the drop-down list.
6. Specify a variable that will get the BW log ID for the process chain after the function executes. You must define the variable before you can use it.
7. Specify a variable that will get the description of the status after the function executes. You must define the variable before you can use it.

3.6.3.119 sap_openhub_set_read_status

Sends the read status for the Open Hub table to SAP NetWeaver BW. A successful read status causes SAP NetWeaver BW to delete the data from the Open Hub Destination table.

Syntax

```
sap_openhub_set_read_status('<datastore>','<destination>',
<status>,<$returntxt>)
```

Where

`<datastore>`	Specifies the datastore name. You can specify either a constant string or a substitution parameter. The data type is varchar(256).

	Note The maximum length depends on the Data Services repository type. For most repository types the maximum length is 256, for MySQL the length is 64, and for MS SQL server the length is 128.
`<destination>`	Specifies the Open Hub Destination in the BW system. You can specify either a constant string or a substitution parameter. The data type is varchar(30).
`<status>`	Specifies the read status. Possible values are either a variable or one of the following string constants: ● 'X' for Read Successful ● Any other value indicates that the Read failed. The data type is varchar(1).
`<$returntxt>`	(Optional) Specifies a variable to return the status log of the function. The required variable data type is varchar, and you can define the length you want for this variable.

Return value

varchar(1)

Returns one of the following values.

Return value	Description
S	Success
E	Error

Example

The following sample script commands sends the status of the Open Hub table read to the BW system and prints the status.

```
$status = sap_openhub_set_read_status('BR9", 'PS_BOOK_5', 'X', $returntxt):
print ('Status: ' || $status);
```

3.6.3.120 search_replace

Performs a simple search and replace based on a string value, word value, or an entire field. You can specify search and replace values with an internal table, an existing external table or file, or with a custom SQL command.

In all cases, the search and replace values are loaded into memory to optimize performance while performing the operation.

Note

We recommend that you use search_replace as a function call in the query output and not as a mapping in a column. By using it as a function call, you can use the wizard interface to fill in the function's parameters, and you can return to the wizard at any time to change the parameters. This method also allows you to create multiple output columns when multiple input expressions are used. Using the function in a script or regular mapping is possible, but the syntax can be hard to read and difficult to maintain.

Below is the function syntax as a reference. The function wizard is explained in the next section.

Syntax

```
search_replace([<sr_table_spec>,<search_column>,<replace_column>],<sr_type>,
[<case_sensitivity>],[<default_replace_value>],[<input_column_list>],
[<output_column_list>],[<output_variable_list>]) SET (...)
```

Where

<sr_table_spec>	A constant string that specifies the search and replace table or file. It has three possible valid forms: **<Datastore.owner.table>** — Specifies a database table containing the search and replace values. **<Fileformat.filename>** — Specifies a fixed or delimited file containing the search and replace values. **NULL** — Used when the search and replace is performed using custom SQL or an internal table defined in the SET options.
<search_column>	Specifies the column name in the table or file containing the search values. If **<sr_table_spec>** is an internal table, this should be set to **NULL**. The data contained in this column must be of the varchar type.
<replace_column>	Specifies the column name in the **<sr_table_spec>** table or file containing the replacement values. If **<sr_table_spec>** is an internal table, this should be set to **NULL**. All data contained in this column must be of the varchar type.
<sr_type>	A string that specifies the type of search and replace operation to perform. It has three possible values: **'SR_FIELD'** — Matches the entire contents of the search field, and replaces the entire contents of the search field.

<table>
<tr><th></th><th></th></tr>
<tr><td></td><td><p><code>'SR_WORD'</code> — Replaces only the word that matches the search value. Any unmatched data remains in the search field. A word is defined as a set of characters set apart by whitespace.</p><p><code>'SR_STRING'</code> — Replaces only a specific substring of characters found next to or between other characters in the search field. Any unmatched data remains in the search field.</p></td></tr>
<tr><td><code>&lt;case_sensitivity&gt;</code></td><td><p>A value that indicates whether or not the search and replace operation is case sensitive.</p><p><code>yes</code> — case sensitive</p><p><code>no</code> — not case sensitive</p></td></tr>
<tr><td><code>&lt;default_replace_value&gt;</code></td><td><p>A varchar that specifies the default replacement value if the search value is not found.</p><p>This applies only when <code>&lt;sr_type&gt;</code> is set to <code>'SR_FIELD'</code>.</p></td></tr>
<tr><td><code>&lt;input_column_list&gt;</code></td><td>A varchar that specifies a comma-separated list of input expressions on which the search and replace operation should be performed.</td></tr>
<tr><td><code>&lt;output_column_list&gt;</code></td><td>A varchar that specifies a comma-separated list of output columns.</td></tr>
<tr><td><code>&lt;output_variable_list&gt;</code></td><td><p>A varchar that specifies a comma-separated list of output variables.</p><p>When more than one output column is specified in <code>&lt;output_column_list&gt;</code>, the output variables are used to receive output returns. Variables and output columns are matched by position.</p><p>This parameter is optional except when more than one output column is specified in <code>&lt;output_column_list&gt;</code>.</p></td></tr>
<tr><td>SET options</td><td><p>Specifies custom SQL or search and replace values in XML format.</p><p>For an internal search and replace table:</p><pre>SET (
"internal_table"=
'<?xml version="1.0" encoding="UTF-8"?>
   <searchTable>
      <item>
         <Search><value></Search>
         <Replace><value></Replace>
      </item>
      ...
      ...
   </searchTable>
')</pre><p>For custom SQL:</p><pre>SET (
"external_custom_sql"=
'<?xml version="1.0" encoding="UTF-8"?>
   <database_datastore&gt;</pre></td></tr>
</table>

```
      <datastore_name>
   </database_datastore>
   <search_column>
      <search_column_name>
   </search_column>
   <replace_column>
      <replace_column_name>
   </replace_column>
   <SQLText>
      <custom_sql_string>
   </SQLText>
')
```

Return value

varchar

Example

Search for Mr in input_column and replace with M to output_column using an internal search and replace table.

```
search_replace(NULL,'SR_STRING',1,,'input_column','output_column',) SET (
"internal_table"='<?xml version="1.0" encoding="UTF-8"?>
   <searchTable>
      <item>
         <Search>Mr</Search>
         <Replace>M</Replace>
      </item>
   </searchTable>
)
```

Note

The search_replace function wizard makes it easy to select search and replace columns, and if needed, define search terms and replacement values.

3.6.3.120.1 To define a search_replace function

1. Right-click the output schema of a Query transform, and click *New Function Call*.
2. Choose the *String Functions* category, and click *search_replace*.
3. Click *Next*.
   The search_replace function wizard opens.
4. Select the search type to perform.

Search type	Description
Full string	Matches and replaces the entire contents of the input expression.
Word	Replaces only the word that matches the search value. Any unmatched data is not modified. A word is defined as a set of characters set apart by whitespace.

Search type	Description
Substring	Replaces only a specific substring of characters found next to or between other characters in the input expression. Any unmatched data is not modified. For a substring search, the search values are ordered by length from largest to smallest, and each substring is only substituted once.

5. If you want to ignore casing differences between the specified search values and the values from the input expressions, uncheck *Case sensitive*.
6. Define the input expressions to search. You can select a column name from the input schema, or you can use any expression that uses one or more input columns.
7. Define the output column names and lengths for the replacement values. You can rename the output column by selecting the value and pressing **F2**.
8. Configure the search and replace table.
   - For an internal table, select *Internal* and specify search values and corresponding replacement values. The values for an internal table are stored in the metadata repository as part of the function definition.
   - For an external relational table, select *External*. Specify a valid datastore or file format as the Source, and select the columns containing the search and replacement values.
   - For custom SQL, select *Custom SQL*. Choose the source datastore, and specify the columns containing the search and replacement values. Define the custom SQL text to run.
9. Specify the default replacement value for any rows that do not match the defined search values. If the replacement value is a fixed string, you must enclose it in single quotes.

   > Note
   >
   > The default replacement value applies only for a full string search, not for word or substring searches.

   If no default replacement value is defined, the original values are preserved; no replacements are made for expressions that were not found.
10. If you want to force the search_replace function to execute in a separate process, check *Run as a separate process*. The search and replace table is always loaded into memory, so for large tables, running this function as a separate process can improve performance.

> Example
>
> Case-sensitive substring search and replace
>
> In this example, the longest search value is replaced first. As a result, the substring AUS does not become AUSA, but is correctly replaced with Australia.
>
> Internal Search and Replace table:
>
> | Search value | Replace value |
> |---|---|
> | US | USA |
> | AUS | Australia |
>
> Search and Replace results:
>
> | Value on input | Value on output |
> |---|---|
> | California, US | California, USA |
> | Melbourne, AUS | Melbourne, Australia |

Related Information

Performance Optimization Guide: Distributing Data Flow Execution, Splitting a data flow into sub data flows, Run as a separate process option [page 2153]

3.6.3.121 set_cdc_checkpoint

Sets a check-point in a data flow for a Microsoft SQL Server changed-data-capture (CDC method) job. Use for data flows that run in a WHILE loop to retrieve changed data for each iteration of the loop. Call this function for all the datastores used in all the data flows of the job.

Syntax

```
set_cdc_checkpoint(<datastore>)
```

Return value

int

Returns 1 if successful, otherwise 0.

Where

<datastore>	The name of the CDC-enabled datastore containing the tables used to obtain the changes.

Example

```
set_cdc_checkpoint('MyCdcSource');
```

3.6.3.122 set_env

Sets a system environment variable to a specified value for the duration of a job.

Syntax

```
set_env('<variable_name>', <variable_value>)
```

Return value

int

Returns 1 if successful, otherwise, 0.

Where

<variable_name>	The name of an environment variable. The name must be surrounded by single quotes.
<variable_value>	The value you want assigned to the environment variable. If the value is text, you must surround the text by single quotes. Data Services only sets the variable to this value for the duration of the current job.

Use the get_env and set_env functions to set and retrieve variables across operations in a job.

Example

Function	Results

`set_env('TMP','C:\Data Services`
`\Temp')`
	Sets the value of the TMP environment variable to "C:\Data Services\Temp" and returns a 1.

Related Information

get_env [page 1568]

3.6.3.123 sleep

Suspends the execution of the calling data flow or work flow.

Syntax

```
sleep(<num_millisecs>)
```

Return Value

int

Returns '1', always.

Where

<num_millisecs>	The number of milliseconds to "sleep".

details

Calling this function causes the thread that executes this function to halt operations for the given number of milliseconds. To force a job to halt operations (until a condition becomes true), call this function in a work flow, not in a data flow.

Example

The following example invokes sleep for one second when a file exists in a directory called 'c'.

```
while (file_exists('c:/temp.msg') == 0)
begin
sleep(1000);
end
```

3.6.3.124 soundex

Encodes the input string using the soundex algorithm and returns a string. Use this function when you want to push-down to the database-level. Results may vary when you push-down to different database types.

Syntax

```
soundex(<input_str>)
```

Return Value

varchar(4)

Returns a string containing the soundex encoding of the input string. The return string length is always four characters.

Where

`<input_str>`	The source string that will be encoded.

Details

Only use this function for input strings in English. Non-English characters are ignored.

Any invalid leading characters in the input string are ignored.

If an input string cannot be encoded, then '0000' is returned.

Example

Function	Result
`Print(soundex('Hello');`	Prints the soundex of the word "Hello."
`$VAR=soundex(emp_name);`	Returns the soundex encoding for the string stored in the variable emp_name and then assigns it to $VAR
`$VAR=soundex('1234567');`	Returns '0000' because the input data is numeric.

3.6.3.125 sql

Runs a SQL operation against tables in the specified database.

Syntax

```
sql(<datastore, sql_command>)
```

Return value

varchar(1020)

Returns the first 1020 characters from the query's output. Typically, if `<sql_command>` is a SELECT statement, the return value is the first row value of the first column. If `<sql_command>` is not a SELECT statement, the return value is typically NULL. You must remember this if you assign the value returned to a variable.

Where

`<datastore>`	A string containing the name of the datastore where the tables involved in the SQL operation reside. This name is the name you specified when you created the datastore in Data Services. Include this string in single quotation marks.
`<sql_command>`	A string containing the text of the SQL command to execute. This string must be enclosed in single quotation marks ('). If the string contains quoted values, the internal quotation marks must be single quotation marks preceded by the escape character, backslash (\\). Data Services makes column and table names uppercase when sending the **`<sql_command>`** to Oracle to resolve. To specify a lowercase column or table name from an Oracle database, enclose the name with double quotation marks (").

Example

Function	Results

`sql('source_ds',`
`'SELECT EmpID FROM Emp WHERE sal > 100000')`
	Runs the SQL command against the database connected through the datastore `source_ds`.

`sql('source_ds','SELECT customer.LastName`
`FROM customer`
`WHERE customer.State = \'CA\'')`
	Returns the last names of the customers in California from the customer table. Note that the quotation marks around the state value require a backslash to indicate that the single quotation marks are considered part of the **`<sql_command>`** string.

`sql('oracle_ds', ('SELECT "start_time-`
`stamp"`
`FROM "status_table"`
`WHERE "extract_name" = \'DF_RecoverDim\'`
`AND "stop_timestamp" = NULL')`
	Returns the timestamp from a status table for the completed data flow. An Oracle datastore requires double quotation marks around the lowercase column and table names.

`$start_date=sql('warehouse_ds',`
`'SELECT finish_timestamp`
`FROM time_table`
`WHERE table_name= "Component_Orders" ');`
	In this script example, because the function returns a varchar value, it is not possible to assign the return value to a date variable directly. Modify your statement to the next example.

`$temp_char=sql('warehouse_ds',`
`'SELECT finish_timestamp FROM time_table`
`WHERE table_name= "Component_Orders" ');`
`$start_date=to_date($temp_char,'dd-mon-`
`yyyy');`
	This script example assumes the database returns the date in `dd-mon-yyyy` format. If you are unsure of the format the database returns, then force it to return the date in a specific format by doing a conversion. To accomplish this, use the to_char function in Oracle or the convert function in MS SQL.

3.6.3.126 sqrt

Returns the square root of the given expression.

Syntax

```
sqrt(num)
```

Return Value

Float

Where

<num>	The number for which you want the square root.

Details

Return value is NULL if the input is negative.

Example

Function	Results
`sqrt(625.25));`	25.005000

3.6.3.127 smtp_to

Captures the specified number of lines in the trace and error logs, packages the information into an e-mail, and sends it to the recipient(s) via an SMTP server. This function is typically used in a script, for example in a conditional clause, while loop, or try-catch block.

Syntax

```
smtp_to(<recipients_list>, <subject>, <message>, <number_of_trace_lines>,
<number_of_error_lines>)
```

Return value

int

Returns 0 if function succeeds. Returns a non-zero integer if function fails.

Where

<recipients_list>	A string containing one or more recipient e-mail addresses separated by commas (,). This string cannot be empty and must contain valid, qualified e-mail address information.
<subject>	A string containing the subject of the e-mail. This string can be empty.
<message>	A string containing the e-mail message. This string can be empty.
<number_of_trace> **<_lines>**	The number of lines from the end of the trace log file to append to the end of the e-mail. This input cannot be empty.
<number_of_error> **<_lines>**	The number of lines from the end of the error log file to append at the end of the e-mail. This input cannot be empty.

Example

smtp_to function

Function	Results
`$myvar = smtp_to('admin@company.com', 'Out of memory error in the SalesFact job. Please fix the error before running recovery job.',' ',10, 10)`	The message goes to one recipient, admin@company.com.
`$myvar = smtp_to('admin@company.com, manager@company.com', 'Out of memory error:' || systime(), 'Out of memory error while running the data flow:' || $dataflow_name || ' in the job:' || $job_name || '.', 10, 20)`	The message goes to two recipients, admin@company.com and manager@company.com. The job name and data flow names are included in the text of the message as variables. Note that Data Services trims blank spaces from the end of strings; this example includes a blank on the beginning of the next string. You can also concatenate a string with a single blank.

<table>
<tr><th>Function</th><th>Results</th></tr>
<tr><td></td><td>In the script, type <code>$a = ;</code> where <code>$a</code> is the local integer variable defined in the work flow. Put the cursor just ahead of the semicolon before clicking the <em>Functions</em> button to construct a smtp_to statement.</td></tr>
<tr><td><code>$address_list = 'nkumar@businessobjects.com,pkulkarn@businessobjects.com';</code>
<code>$subject_text = 'Testing SMTP job smtp011';</code>
<code>$message_text = 'The job smtp011 has the following trace lines & errors';</code>
<code>$trace_log = 8;</code>
<code>$error_log = 19;</code>
<code>$smtp_output = 9;</code>
<code>print ('before execution :- ' || $smtp_output);</code>
<code>$smtp_output = smtp_to($address_list,$subject_text,$message_text,</code>
<code>$trace_log,$error_log);</code>
<code>print ('after execution :- ' || $smtp_output);</code></td><td>The smtp_to function also supports global variables. This example from a script shows the smtp_to function substituting the values from global variables.</td></tr>
<tr><td>Create a custom function my_smtp that contains the following:
<code>begin return(smtp_to('nkumar@ness-gsg.com','Testing job smtp012','Test message of smtp012 job',4,5)); End</code>
Use the custom function my_smtp in a script as follows:
<code>print('before smtp :- '); my_smtp( );</code>
<code>print('after smtp :- ');</code></td><td>You can use smtp_to in any user-defined custom function and invoke it in a script.</td></tr>
</table>

> **Note**
>
> The smtp_to function does not support nicknames.

3.6.3.127.1 To define and enable the smtp_to function

1. In the function editor, click the *System* Function Category.
2. Click the *smtp_to* function name.
3. Click *Next*.
4. Define the input parameters as described in the *Where* table previously in this function description.
5. Click *Finish*.
6. Open the Data Services Server Manager:

- In Windows, click *Start* > *Programs* > *SAP Data Services* **<x.x>** > *Data Services Server Manager*.
- In UNIX, run the Server Manager by entering:

```
$ cd $LINK_DIR/bin/
```

```
$ . ./al_env.sh
```

```
$ ./svrcfg
```

7. The Data Services Server Manager Utility window opens.
8. Enter the SMTP server settings:
   - *Server name*: Type the name or IP address of the SMTP server (for example mail.company.com)
   - *Sender*: Type the e-mail address that will appear in the *From* field of the e-mail.
9. In the Server Manager window, click *Apply*.

3.6.3.128 substr

Returns a specific portion of a string starting at a given point in the string.

Syntax

```
substr(<input_string>, <start>, <length>)
```

Return value

varchar

The modified string. The return data type is the **<input_string>**. If the length is a constant, then it is a varchar of the given length.

Where

<input_string>	The string to be modified.
<start>	The position of the first character in the new string. The function counts characters from the beginning of **<input_string>**. In normal data flows, the first character is position number 1. If **<start>** is 0, the new string begins with the first character (position 1).

	If <start> is negative, the function counts characters from the end of <input_string>. The new string begins with the character in that position from the end of the string. If <start> is greater than the number of characters in <input_string>, the function returns NULL. If <length> is 0 or negative and <start> is not greater than the number of characters in <input_string>, the function returns empty string.
<length>	The number of characters in the resulting string. If <length >is 0 or negative, the function returns an empty string. If <length> is greater than the number of characters remaining in <input_string> after <start> , the function returns only the remaining characters. The function keeps the trailing blanks in the remaining <input_string> after <start>.

Example

Function	Results
`substr('94025-3373', 1, 5)`	`'94025'`
`substr('94025-3373', 7, 4)`	`'3373'`
`substr('94025', 7, 4)`	`NULL`
`substr('Dr. Schultz', 4, 18)`	`'Schultz'`
`substr('San Francisco, CA',-4, 18)`	`', CA'`

3.6.3.129 sum

Calculates the sum of a given set of values.

Syntax

```
sum(<value_list>)
```

Return value

decimal, double, int, or real

The total of the values. The return type is the same as the values in **`<value_list>`**.

Where

`<value_list>`	The source values to sum.

Example

To calculate the sum of values in the salary column of a table, use the sum function in a query:

- In the *Mapping* tab of the query editor, enter:

  ```
  sum(SALARY)
  ```

- In the *Group By* tab in the query editor, specify the columns for which you want to find the total salary, such as the department column. For each unique set of values in the group by list, such as each unique department, Data Services calculates the sum of the salary.

3.6.3.130 sysdate

Returns the current date as listed by the Job Server's operating system.

Note

The value that the sysdate function returns is a datetime value. Internally Data Services reads both the date and the time when it runs a sysdate function. The data that is used by the job depends on the data type of a particular column. For example, if the data type of a column in a query is `date`, Data Services only uses the date for calculations. The time data is ignored. If you change the data type to `datetime`, both a date and a time are used.

Syntax

```
sysdate()
```

Return value

date

Today's date.

Example

Function	Results
`isweekend(sysdate())`	Tests whether today is a Saturday or Sunday.
`to_char(sysdate(), 'yyyy.mm.dd')`	Converts the sysdate function's `datetime` value to a string that displays only the date. For example, you can use this to exclude part of the `datetime` data by only providing a format for the data you want to display in a report. To convert a `datetime` value to a string containing only the date, use this expression and change the column's data type to `varchar`.

3.6.3.131 system_user_name

Returns the user name used to log into the Job Server's operating system.

Syntax

```
system_user_name()
```

Return Value

varchar

Example

```
print('Starting execution of Job: [job_name()] as user: [system_user_name()]');
```

3.6.3.132 systime

Returns the current time as listed by the Job Server's operating system.

Syntax

```
systime()
```

Return value

time

The current time.

Example

Function	Results
`$timestamp = sql('my_datastore',('UPDATE status_table SET job_start_time = \' '|| to_char(systime(),'hh24:mi:ss.ff'))||'\'');`	This expression updates the `job_start_time` column of the `status_table` with the current time. It also formats the time data.
`to_char(systime(),'hh24:mi:ss.ff')`	Trims date data from the systime() function in cases where it is added by default. Set the column that contains this expression to the data type `varchar`. The data type for a column that calls the systime() function should be `time`. If the data type is set to `datetime`, Data Services will add the default date for the `datetime` data type (1900:01:01) because systime() does not read dates.

3.6.3.133 table_attribute

Retrieves the value of the specified table attribute.

Syntax

```
table_attribute(<table_name, attribute_name>)
```

Return Value

varchar

The value of the table attribute. If the specified attribute does not exist, NULL is returned.

where

<table_name>	Use the following format: 'datastore.owner.table'. If a valid table by this name does not exist, NULL is returned. This parameter is case sensitive.
<attribute_name>	The name of a table attribute. Valid attributes for a table are listed in the table's Properties window. This parameter is case sensitive.

Example

Function	Result
table_attribute('mssql.awez.CUSTOMER', 'Number_Of_Inserts')	'1788'

3.6.3.134 to_char

Converts a date or numeric data to a string. It supports the Oracle 9i timestamp data type up to 9 digits precision for sub-seconds.

Syntax

```
to_char(<date or numeric_expression>, <format>)
```

Return value

varchar

Where

A formatted string describing **<numeric_expression>**.

<numeric expression>	The source int, real, double or decimal value.
<format>	A string indicating the format of the generated string. Note Please provide format to ensure correct results. Choose from the following codes:

Format	Description	Example
9	Number (suppress leading/trailing zeros) Includes a leading - for negative numbers or one leading space for pos no's.	to_char(123,'9999') = ' 123'
0	Number Including leading/trailing zeros.	to_char(123,'09999') = ' 0123' to_char(123,'9999D.00') = ' 123.00'
D<.|,>	Position of decimal point followed by character to be used as decimal separator. Currently only dot(.) and comma(,) are supported as decimal points. Currently only dot(.) and comma(,) are supported as decimal points.	to_char(12.34,'99D.99') = ' 12.34'
G<.|,| space >	Position of group separator followed by character to be used as group separator. Currently only dot(.), comma(,) and space(' ') are supported as group separator.	to_char(1234,' 9G,999') = ' 1,234'
x	String containing unsigned hexadecimal integer, using "abcdef". Output is not padded if number is not 2 bytes long.	to_char(123,'xx') = ' 7b' to_char(12,'x') = ' c'
X	String containing unsigned hexadecimal integer, using "ABCDEF". Output is not padded if number is not 2 byte long.	to_char(123,'XX') = ' 7B' to_char(12,'X') = ' C'
O	String containing unsigned octal integer. This option is "not" case sensitive. Output is not padded if number is not 2 bytes long.	to_char(12,'oo') = ' 14' to_char(1,'o') = ' 1'

A formatted string describing `<date>`.

`<date>`	The source date, time, or datetime value.
`<format>`	A string indicating the format of the generated string. Choose from the following codes: DD: 2-digit day of the month MM: 2-digit month MONTH: Full name of month MON: 3-character name of month YY: 2-digit year YYYY: 4-digit year HH24: 2-digit hour of the day (00-23)

	MI: 2-digit minute (00-59) SS: 2-digit second (00-59) FF: Up to 9-digit sub-seconds

Other values included in **<format>** appear in the result unchanged.

Example

Function	Results

`to_char(call_date,`
`'dd-mon-yy hh24:mi:ss.ff')`
	The date value from the `call_date` column formatted as a string, such as:
	`28-FEB-97 13:45:23.32`

The hyphens and spaces in **<format>** in the example are reproduced in the result; all the other characters are recognized as part of a parameter string in the table above and substituted with appropriate current values.

Related Information

timestamp [page 1037]

3.6.3.135 to_date

The to_date function converts a string to a date based on the input format.

If the input string has more characters than the format string, the extra characters in the input string will be ignored and will be initialized to the default value.

For example, to_date('10.02.2007 13:25:45', 'DD.MM.YYYY') will convert the date to 10.02.2007 00.00.00. The time part in the input string will be ignored and initialized to 0.

This function also supports the Oracle 9i `timestamp` data type. Its precision allows up to 9 digits for sub-seconds.

Syntax

```
to_date(<input_string>, <format>)
```

Return value

date, time, or datetime

A date, time, or both representing the original string.

Where

<input_string>	The source string.
<format>	A string indicating the format of the source string. Choose from the following codes: DD: 2-digit day of the month MM: 2-digit month MONTH: Full name of month MON: 3-character name of month YY: 2-digit year YYYY: 4-digit year HH24: 2-digit hour of the day (00-23) MI: 2-digit minute (00-59) SS: 2-digit second (00-59) FF: Up to 9-digit sub-seconds

Example

Function	Results
`to_date('Jan 8, 1968', 'mon dd, yyyy')`	`1968.01.08` stored as a date.

Related Information

timestamp [page 1037]

3.6.3.136 to_decimal

Converts a varchar to a decimal.

Syntax

```
to_decimal(<in_str, decimal_sep, thousand_sep, scale>)
```

Return Value

decimal

Uses a precision of 28 and the given scale.

Where

<in_str>	The number string. Null implies a NULL return.
<decimal_sep>	The character that separates the decimal component from the whole number component.
<thousand_sep>	The character that separates thousands from hundreds in the whole number component.
<scale>	The number of digits to the right of the decimal point in the returned value.

Details

Takes a string that represents a number and converts it to a decimal. If the input string is invalid, a 0 is returned.

Example

Function	Result
`to_decimal('99,567.99', '.', ',', 3)`	`99567.990`

3.6.3.137 to_decimal_ext

The to_decimal_ext function supports the use of DECIMAL data types with up to 96 precision. This function converts a varchar to a decimal and includes precision as a parameter.

Syntax

```
to_decimal_ext(<in_str, decimal_sep, thousand_sep, precision, scale>)
```

Return Value

decimal

Uses the given precision and scale.

Where

`<in_str>`	The number string. Null implies a NULL return.
`<decimal_sep>`	The character that separates the decimal component from the whole number component.
`<thousand_sep>`	The character that separates thousands from hundreds in the whole number component.
`<precision>`	The total number of digits in the returned value.
`<scale>`	The number of digits to the right of the decimal point in the returned value.

Details

Takes a string that represents a number and converts it to a decimal. Returns 0 if the input string is invalid.

Example

Function	Result
`to_decimal_ext('99,567.99', '.', ',', 38,3)`	`99567.990`

3.6.3.138 total_rows

Returns the number of rows in a particular table in a datastore. This function can be used with any type of datastore.

Syntax

```
total_rows(<datastore.owner.table_name>)
```

or for a memory datastore:

```
total_rows(<datastore..table_name>)
```

Return value

int

The number of rows in the table.

Where

<datastore>	The name of the datastore containing the table.
<owner>	The name of the datastore owner. Not used for memory tables.
<table_name>	The name of the database table or memory table containing the rows you want to count.

Example

Function	Results
`total_rows (ora_ds.scott.emp_table)`	Retrieves the total number of rows from an Oracle table.
`total_rows (mem_ds..bigtable)`	Retrieves the total number of rows from a memory table.

3.6.3.139 trunc

Truncates a given number to the specified precision, without rounding the value.

Syntax

```
trunc(<num1>, <precision>)
```

Return value

decimal, double, int, or real

The truncated number. The return type is the same as the original number, **<num1>**.

Where

<num1>	The source number.
<precision>	An integer indicating the number of decimals in the result. If **<precision>** is negative, digits to the left of the decimal point are truncated and the value is padded with zeros.

Example

Function	Results
`trunc(120.12345, 2)`	120.12
`trunc(120.12999, 2)`	120.12
`trunc(180, -2)`	100
`trunc(120.123, 5)`	120.12300

3.6.3.140 truncate_table

Allows you to explicitly expunge data from a memory table or truncate physical files used for a persistent cache table. With regard to memory tables, this function provides finer control than the active job has over your data and memory usage. Use this function with memory tables and persistent cache tables.

Syntax

```
trunc(<ds..tab_name>)
```

Return value

int

The return value is always 1.

Where

`<ds>`	The datastore containing the memory table or persistent cache table.
`<tab_name>`	The name of the memory table from which you want to expunge data or the name of the persistent cache table from which you want to truncate physical files.

Example

Function	Results
`truncate_table (ds..bigtable)`	Truncates rows from the memory table or persistent_cache table.

Usage scenarios

1. A data flow in your job creates a persistent cache table which can be used by subsequent data flows within the job (through the lookup_ext function, for example).
   You can add a custom function directing the truncate_table() function to remove the persistent cache table after running all data flows within the job.
2. Create a job that includes a script to clean up all unused persistent cache tables. When run, the job would truncate your physical files and free disk space.

3.6.3.141 upper

Changes the characters in a string to uppercase.

Syntax

```
upper(<value>,'locale')
```

Return value

varchar

The uppercase string. The return type is the same as **<value>**. Any characters that are not letters are left unchanged.

Where

<value>	The string to be modified.
<locale>	Optional parameter that converts the string to the specified locale. Note ISO 639 language code and ISO 3166 country code formats are supported.

Example

Function	Results
`upper('Accounting101')`	`'ACCOUNTING101'`
`upper(substr(LastName,1,1))|lower(substr(LastName,2,LENGTH(LastName)))`	The value in column `LastName` with the first letter uppercase and the rest of the value lowercase. Note that this example does not account for two-word last names.
`upper(LastName, 'tr')`	The value in column `LastName` is converted to all lowercase. It is also converted to the Turkish locale, using the ISO 639 language code.

Related Information

ISO 639 language list

ISO 3166 Country Code list

3.6.3.142 varchar_to_long

Converts a data type value of a given column from varchar to long.

Syntax

```
varchar_to_long(<column_name>)
```

Return value

long

Where

`<column_name>`	The name of the table column for which you want to convert a data type from varchar to long.

Related Information

Designer Guide: XML extraction and parsing for columns, Scenario 2 [page 366]

3.6.3.143 wait_for_file

This function looks for the specified file pattern in the file system. If it does not find the file(s), it waits for the specified timeout period, polling for the file(s) at every polling interval. The value specified in poll_interval determines how often to poll for the file pattern until timeout is reached. After timeout, the job stops, and polling for the file ceases.

Syntax

```
wait_for_file ( file_name_pattern, timeout, poll_interval,max_match, file_name_list, list_size, list_separator)
```

Return Values

int

Values are:

- 0 - No file matched.
- 1 - At least one file is matched.
- -1 - Timed out.
- -2 - At least one input value is illegal.

Where

`<file_name_pattern>`	The file name and path, relative to where the Job Server is running. It can be an absolute or relative path. File name can contain wildcard characters.
`<timeout>`	Wait timeout in milliseconds. If timeout is 0, then the function does not block. If timeout is -1, then the function will wait indefinitely for at least one file to exist that matches the file pattern. Any other negative value is illegal. On a computer where millisecond timing accuracy is not available, timeout is rounded up to the nearest legal value available on that system.
`<poll_interval>`	Polling interval in milliseconds to look for the existence of file(s). On a computer where millisecond timing accuracy is not available, the polling interval is rounded up to the nearest legal value available on that system. If the poll interval exceeds the timeout value then, it will be rounded up to timeout value.
`<max_match >`	Optional. Specifies the maximum number of matched file names that function should return. The default value is 0. -1 specifies all the matched file names.
`< file_name_list >`	Optional. Output varchar variable that returns the list of matched file names. Order of the file names in the list is determined the way operating system returns the file names.
`< list_size>`	Optional. Output integer variable that returns the list size.
`<list_separator>`	Optional. File name list separator character(s). Default value is comma (,).

Details:

This function waits a maximum of up to timeout interval for at least one file to exist that matches the pattern. Poll interval determines how often to poll for files.

Example

This function is used in a script at the beginning of a job. A job will suspend until a file is present, as shown in the following business use case example:

During the night, an external process puts source files in a file system that Data Services can access. Usually this process is finished at 1:00 AM, but it can be later. You schedule the job to start at 1:00 AM, but in the first step of the job use a script that checks for the existence of the last file. If the file does not exist, the job will wait for some time and try again later. Once the file is present, the job will continue. A timeout needs to be set to stop the job when the file is still not present at 9:00 AM in the morning.

3.6.3.144 week_in_month

Determines the week in the month in which the given date falls.

This function considers the first week of the month to be first seven days. The day of the week is ignored when calculating the weeks.

Syntax

```
week_in_month(<date1>)
```

Return value

int

The number from 1 to 5 that represents which week in the month that **<date1>** occurs.

Where

<date1>	The source date.

Example

Function	Results
`week_in_month(to_date('Jan 22, 1997', 'mon dd, yyyy'))`	4
`week_in_month(to_date('Jan 21, 1997', 'mon dd, yyyy'))`	3

3.6.3.145 week_in_year

Returns the week in the year in which the given date falls.

This function returns the week in the year in two ways:

- 'WW' - Absolute week number of the given date.
- 'IW' - ISO week number of the given date.

Syntax

```
week_in_year(inputdate,weektype)
```

Return value

int

Where

<inputdate>	The source date
<weektype>	Optional. The values are 'WW', or 'IW'. Default value is "WW".

Description

The number from 1 to 53 that represents the week number in a year. This function considers the first week of the year to be the first seven days while determining the absolute week number. Under the ISO standard, a week always begins on a Monday, and ends on a Sunday. The first week of a year is that week which contains the first Thursday of the year. An ISO week number may be between 1 and 53. Under the ISO standard, week 1 will always have at least 4 days. If 1-Jan falls on a Friday, Saturday, or Sunday, the first few days of the year are defined as being in the last (52nd or 53rd) week of the previous year.

Example

Some business applications use week numbers to categorize dates. For example, a business may report sales amounts by week, and identify each period as "9912", representing the 12th week of 1999. (An ISO week is more meaningful than an absolute week for such a purpose.)

Following are more example results for week_in_year applied to three different input dates:

Function	Results
`week_in_year(to_date('Jan 01, 2001','mon dd, yyyy'))`	1
`week_in_year(to_date('2005.01.01', 'yyyy.mm.dd'),'WW')`	1

Function	Results
`week_in_year(to_date('2005.01.01', 'yyyy.mm.dd'),'IW')`	`53`

3.6.3.146 WL_GetKeyValue

Returns the value of a given keyword in Web log search strings.

Syntax

```
WL_GetKeyValue(<string>, <keyword>)
```

Example

If you search for **`BusinessObjects`** on Google, the following appears in a Web log:

```
GET "http://www.google.com/search?hl=en&lr=&safe=off&q=BusinessObjects&btnG=Google
+Search"
WL_GetKeyValue('http://www.google.com/search?
hl=en&lr=&safe=off&q=BusinessObjects&btnG=Google+Search','q') returns
'BusinessObjects'.
```

3.6.3.147 word

Returns one word out of a string.

Syntax

```
word(<input_string>, <word_num>)
```

Return value

varchar

A string containing the indicated word. The return type is the same as **`<input_string>`**.

Where

<input_string>	The source string.
<word_num>	A nonnegative integer specifying the index of the target word in the string. The first word in a string is word number 1. If **<word_num>** is 0 or greater than the number of words in **<input_string>**, then the word function returns a NULL string.

A word is defined to be any string of consecutive non-white space characters terminated by white space, or the beginning and end of **<input_string>**. White space characters are the following:

- Space
- Horizontal or vertical tab
- Newline
- Linefeed

Example

Function	Results
`word('Accounting Department', 1)`	`'Accounting'`
`word('Accounting', 1)`	`'Accounting'`
`word('Accounting', 2)`	`NULL`

3.6.3.148 word_ext

Returns the word identified by its position in a delimited string.

This function is useful for parsing Web log URLs or file names.

Syntax

```
word_ext(<string>, <word_num, separator(s)>)
```

Return value

varchar

A string containing the indicated word. Return type is the same as **`<string>`**.

Where

`<string>`	The source string.
`<word_num>`	A nonnegative integer specifying the index of the target word in the string. The first word in a string is word number 1. If **`<word_num>`** is 0 or greater than the number of words in **`<string>`**, then the word function returns a NULL string.
`<separator(s)>`	Any character specified.

A word is defined to be any string of consecutive non-white space characters terminated by white space, or the beginning and end of **`<string>`**. White space characters are the following:

- Space
- Horizontal or vertical tab
- Newline
- Linefeed

Example

Function	Results
`word_ext('www.sap.com',2,'.')`	`'sap'`
`word_ext('www.cs.wisc.edu', -2, '.')`	`'wisc'` A negative word number means count from right to left.
`word_ext('www.cs.wisc.edu', 5, '.')`	`NULL`
`word_ext('aaa+=bbb+=ccc+zz=dd', 4, '+=')`	`'zz'` If 2 separators are specified (+=), the function looks for either one.
`word_ext(',,,,,aaa,,,,bb,,,c ', 2, ',')`	`'bb'` This function skips consecutive delimiters.

3.6.3.149 workflow_name

Returns the name of the current work flow.

Syntax

```
workflow_name()
```

Return Value

varchar

Details

In cases where several work flows enclose this function, the function returns the name of the inner most work flow. If no work flow is found, the function returns the job name.

Example

```
print('Work Flow Name: [workflow_name()]')
```

3.6.3.150 year

Determines the year in which the given date falls.

Syntax

```
year(<date1>)
```

Return value

int

The number that represents the year component of **<date1>**.

Where

<date1>	The source date.

Example

Function	Results
`year(to_date('Jan 22, 1997','mon dd, yyyy'))`	1997
`year(to_date('03/97', 'mm/yy'))`	1997
`year(to_date('03/19', 'mm/yy'))`	2019

3.6.4 Custom functions

You can create your own functions by writing script functions in SAP Data Services scripting language using the smart editor. Saved custom functions appear in the function wizard and the smart editor under the *Custom* category. They also are displayed in the object library under the *Custom Functions* tab. You can edit and delete custom functions from the object library.

In the object library on the *Custom Functions* tab, there are multiple categories of custom functions:

- Custom Functions
- Validation Functions:
  - Imported from Information Steward: These functions were created in Information Steward and imported; they are not editable in Data Services.
  - Locally Created: These are reusable, custom, validation functions created in Data Services.

Consider these guidelines when you create your own functions:

- Functions can call other functions.
- Functions cannot call themselves.
- Functions cannot participate in a cycle of recursive calls. For example, function A cannot call function B, which calls function A.
- Functions return a value.
- Functions can have parameters for input, output, or both. However, data flows cannot pass parameters of type output or input/output.

Before creating a custom function, you must know the input, output, and return values and their data types. The return value is predefined to be `Return`.

Related Information

Scripting Language [page 1709]

3.6.4.1 To create a custom function

1. Choose ▶ *Tools* ▶ *Custom Functions* ◀.
2. In the Custom Function list, right-click and select *New*.

   Alternatively, from the object library, right-click and select New in the Custom Functions tab.
3. Enter the name of the new function.
4. Enter a description for your function.
5. Click *Next* to open the smart editor.

   In the smart editor, you can define the return type, parameter list, and any local variables to be used in the function.
6. In the Variables tab, right-click *Return* and select *Properties...*

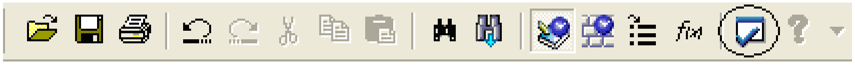
   By default, the return data type is set to int. To change this, select another return data type from the Data type list. Click OK.
7. In the Variables tab, right-click *Parameters* and choose *Insert*.
8. Define parameter properties by choosing a Data type and a Parameter type (Input, Output, or Input/Output).

   > **Note**
   >
   > Data flows cannot pass variable parameters of type output and input/output.
9. Click *OK*.

   Repeat steps 7 - 9 for each parameter required in your function. After you add one parameter, the right-click menu allows you to choose where to insert each new one.

   Use this menu to create, delete, or edit variables and parameters.
10. To define variables used by the function, but not passed outside the function, right-click *Local* and choose *Insert*.
11. Choose a data type in the Variable Properties window and click *OK*.

    Repeat this step for each variable required in your function.
12. Complete the text for your function.
13. Click the *Validate* icon to validate your function.

    If your function contains syntax errors, you will see a listing of those errors in an embedded pane below the editor.
14. To see where the error occurs in the text, double-click on an error.

    The smart editor redraws to show the location of the error.
15. When your function is valid, click *OK* to save the function, and the variables and parameters inside.

    Variables and parameters for an existing custom function are local to each function. Therefore, they are not displayed in the Variables and Parameters window (accessible from ▶ *Tools* ▶ *Variables* ◀). Variables and parameters for custom functions can be viewed in the smart editor library, under the Variables tab, when you edit the custom function.

Related Information

Smart editor [page 1021]

3.6.4.2 To edit an existing function

Choose *Tools* › *Custom Functions*, select the function and select *Next*.

The smart editor opens with the function displayed and the variables and parameters that exist for this function shown under the Variables tab.

Alternatively:

- Go to the Custom Function tab in the object library and double-click the function you want to edit.
- Go to the Custom Function tab in the object library, right-click the function, and select *Edit*.

Related Information

Custom functions [page 1694]
To edit an existing function call in an expression [page 1514]

3.6.4.3 To replicate a custom function

1. From the object library, right-click a custom function and select *Replicate*.

   The Custom Function editor opens with only the Function name field enabled.
2. Enter a new name for the function.

   The name must be valid and different from the original name.
3. Click *Finish*.

   The new custom function appears in the object library.

3.6.4.4 To delete a custom function

Go to the Custom Function tab in the object library, right-click the function, and select *Delete*.

If you delete a function from the list of custom functions, you must remove references to the function from expressions in scripts, conditionals, queries, and other custom functions.

3.6.5 About procedures

Data Services supports the use of stored procedures for DB2, ODBC, Oracle, Microsoft SQL Server, SAP HANA, SAP Sybase SQL Anywhere, SAP Sybase ASE, SAP Sybase IQ, and Teradata databases. You can call stored procedures from the jobs you create and run in Data Services.

3.6.5.1 Overview

A stored procedure is a generic term used to describe an executable object, or a named entity that is stored in a database and can be invoked using input and output parameters. Generally, a stored procedure is one or more precompiled SQL statements. By calling a stored procedure from within Data Services, you can invoke business logic you have already coded thus enabling you to quickly and conveniently develop data extraction and data management tasks. Stored procedures can also be used to:

- Maintain business logic rules and provide a single point of control to ensure rules are accurate and enforced
- Significantly reduce network overhead with client/server applications because:
  - Procedures are stored on the database server
  - Compiled execution plans for procedures are retained in the data dictionary

Data Services supports stored procedures for DB2, ODBC, Oracle, Microsoft SQL Server, SAP HANA, SAP Sybase SQL Anywhere, SAP Sybase ASE, SAP Sybase IQ, and Teradata databases. Data Services also supports stored functions and packages for Oracle databases. Queries, scripts, conditionals, and custom functions can all be configured to include stored procedures, stored functions, and packages.

Stored procedures must exist in the database before you can use the procedure in Data Services. Create a stored procedure in a database using the client tools provided with the database, such as Oracle SQL *Plus. After it is created, a stored procedure can be called by users who have execution privileges for the procedure. After they are imported into Data Services, stored procedures can be used like functions in Data Services jobs.

Stored procedures include parameters. Each parameter has a name, a data type, and a mode (`IN`, `INOUT`, or `OUT`). A stored procedure can use a NULL or default parameter value for its input and can produce more than one output parameter value.

3.6.5.2 Requirements

To use stored procedures with Data Services, the following requirements must be met:

- The client and server versions must match.
- Only user-defined stored procedures can be used. Data Services does not support stored procedures provided by a database system.
- User-defined stored procedures or stored functions must meet the following additional requirements:

- The return type (the data type of the result value) must be a Data Services supported data type, such as `varchar`, `date`, or `int`.

  > ℹ Note
  >
  > This release does not support the long data type in stored procedures.

- The name of the stored procedure—the combination of the datastore name, owner name, and procedure name—must be unique. Data Services only imports the first procedure or function with a particular name. Using Oracle for example, if you have multiple procedures or functions with the same name in a package and want to use all of them, you must rename the procedures, giving each a unique name. A procedure is overloaded when multiple versions exist in a particular package. Overloading is not supported in Data Services.

- Data Services validates the user name specified in the datastore which was used to import the stored procedure. This ensures that the data flow that calls a stored procedure enforces restrictions associated with that procedure, such as:
  - Data input restrictions
  - Execution privileges
- To use stored procedures with Teradata, you must first enable the use of stored procedures within the Teradata ODBC driver options and invoke the stored procedures in the same session mode in which it was enabled.
  For Windows, in the *Teradata ODBC Driver Options* screen used for configuration of the Teradata ODBC driver, check the box *Disable CALL to EXEC Conversion*, set the session mode as desired in the *Session Mode* drop-down list, and click *OK* to save the change. You must invoke the stored procedures in the same session mode.
  For UNIX platforms, edit the file `odbc.ini` to set the option **`Disable CALL To EXEC Conversion=YES`**. Set the session mode option as desired, and save the file. You must invoke the stored procedures in the same session mode.
  For more information about Teradata ODBC driver configuration, see your Teradata ODBC driver configuration documentation.

3.6.5.3 Creating stored procedures in a database

This section provides:

- An example scenario for using a stored procedure
- Tips for creating stored procedures on each database
- Example syntax for DB2, Oracle, SAP HANA, SQL Server, and SAP Sybase ASE databases based on the example scenario

Since syntax varies between databases, the SQL statements needed to create a stored procedure for any scenario vary. The client tools used to pass the SQL statements to the database server also vary. Refer to your database documentation for more detailed descriptions and examples regarding how to create stored procedures.

In the following example, the source database has a stored procedure `Get_emp_rec` that retrieves an employee's name and hire date from the Employee table using a given employee number. The stored procedure takes an input parameter `Emp_number` and returns two values `Emp_name` and `Hire_date` via output parameters.

The schema of the database table Employee is as follows:

Name	Null?	Type
empno	Not Null	Integer
ename	Null	Varchar(20)
job	Null	Varchar(9)
mgr	Null	Integer
hiredate	Null	Date
sal	Null	Decimal(7,2)
comm	Null	Decimal(7,2)
deptno	Null	Integer

3.6.5.3.1 Creating stored procedures in DB2

In DB2, a parameter can be `IN`, `OUT` or `INOUT`. Any SQL data type can be used as a data type of a parameter. User defined data types are not supported.

Example

The DB2 syntax for the stored procedure

```
CREATE PROCEDURE GET_NAME_USING_ID (IN NID INTEGER, OUT outVar varchar(20))
language SQL
reads sql data
BEGIN
select first_name into outVar FROM CONTACT where id = NID;
END
```

`Reads sql data` and `language SQL` are two of the many options that DB2 stored procedures support. Refer to the DB2 documentation for the DB2 stored procedure options and their meaning.

Related Information

Creating stored procedures in a database [page 1698]

3.6.5.3.2 Creating stored procedures in Oracle

Oracle supports both stored procedures and stored functions. Any stored procedure that returns a value is called a stored function. Oracle is the only database to allow return values with data types other than an integer.

In Oracle, stored procedures are created using the `CREATE [OR REPLACE] PROCEDURE` statement, and stored functions are created using the `CREATE [OR REPLACE] FUNCTION` statement. The `OR REPLACE` option allows you to override an existing definition of a procedure or function with the same name.

Oracle supports `IN`, `OUT`, and `INOUT` parameters. An `INOUT` parameter allows you to pass in a parameter, modify it, and return the modified value. You can use the DEFAULT keyword or the assignment operator to give an `IN` parameter a default value. When an `IN` parameter has a default value, you can omit the parameter from the argument list when you call the procedure. If you do specify an argument value in the call, the specified value overrides the default value. `INOUT` and `OUT` parameters must be specified.

Example

Syntax

The Oracle syntax for the stored procedure:

```
CREATE OR REPLACE PROCEDURE
get_emp_rec (Emp_Number IN NUMBER,
Emp_Name OUT VARCHAR, Emp_Hiredate OUT DATE) AS
BEGIN
SELECT ename, hiredate
INTO Emp_Name, Emp_Hiredate
FROM Employee
WHERE empno = Emp_Number;
END;
/
```

In this example, the parameters are declared as IN and OUT, but Oracle also supports an INOUT parameter type, which allows you to pass in a parameter, modify it, and return the modified value. INOUT parameters can take default values. This means that the parameter can be omitted from the actual parameter list when you call the procedure. INOUT and OUT parameters must be specified.

An Oracle package is an encapsulated collection of related program objects (e.g., procedures, functions, variables, constants, cursors, and exceptions) stored together in the database. Data Services allows you to import procedures or functions created within packages and use them the same way as top-level procedures or functions.

Data Services does not support overloading of procedures or functions. Overloading a procedure means creating multiple procedures with the same name in the same package, each taking arguments of a different number or data type. If you have multiple procedures or functions with the same name in the same package (and you want to use all of them), you need to rename the procedures or functions so that they all have distinct names. Otherwise, Data Services will only import the first procedure or function by that name.

Related Information

Creating stored procedures in a database [page 1698]

3.6.5.3.3 Creating stored procedures in SAP HANA

SAP Data Services supports SAP HANA stored procedures with zero, one, or more output parameters.

Data Services supports scalar data types for input and output parameters. Table data types are not supported; if you try to import a procedure with table type, you will receive an error message. Data Services does not support data types such as binary, blob, clob, nclob, or varbinary for SAP HANA procedure parameters.

Procedures can be called from a script or a Query transform as a new function call.

Example

Syntax

The SAP HANA syntax for the stored procedure:

```
CREATE PROCEDURE GET_EMP_REC (IN EMP_NUMBER INTEGER, OUT EMP_NAME VARCHAR(20),
OUT EMP_HIREDATE DATE) AS
BEGIN
     SELECT ENAME, HIREDATE
        INTO EMP_NAME, EMP_HIREDATE
     FROM EMPLOYEE
     WHERE EMPNO = EMP_NUMBER;
END;
```

Limitations

SAP HANA provides limited support of user-defined functions (usually written in L), which can return one or several scalar values. The usage of such functions is limited to the projection list and the GROUP BY clause of an aggregation query on top of an OLAP cube or a column table. These functions are not supported by Data Services.

SAP HANA procedures cannot be called from a WHERE clause.

Related Information

Creating stored procedures in a database [page 1698]

3.6.5.3.4 Creating stored procedures in MS SQL Server or SAP Sybase ASE

In Microsoft SQL Server and SAP Sybase ASE, a stored procedure may have a group number (between 1 and 32767), which is an optional integer used to group procedures of the same name so they can be dropped together with a single `DROP PROCEDURE` statement.

The first procedure you create with a name automatically is assigned group number 1. While you can execute this procedure without specifying the group number, other procedures with the same procedure name must be called with the group number. For example, `execute orderproc;2` and so on. Data Services imports the procedure with the group number 1, i.e., the first procedure, unless you specify the name including the group number, such as `orderproc;2`.

A parameter can be either an input parameter (default) or an output parameter. A parameter can not be both input and output. In the following sample, the first parameter `@Emp_Number` is an input parameter and the other two parameters, `@Emp_Name` and `@Emp_Hiredate`, are output parameters. Output parameters are specified using the keyword `OUTPUT`. Input parameters can take default values (database defaults). If a default value is defined, the procedure can be executed without specifying a value for that parameter. You specify a parameter name with "@" as the first character.

An optional integer return value is automatically added to each stored procedure to store its return status. In the following example, if the `SELECT` statement is successful, "`0`" is returned as the return value. The parameter name of the return value is `RETURN_VALUE`. Microsoft SQL Server and SAP Sybase ASE stored procedures can be used like Oracle stored functions in Data Services because they include this return value.

A user-defined function is a new feature of SQL Server 2000. Data Services supports user-defined stored procedures, but not user-defined functions at this time.

Example

The MS SQL Server and SAP Sybase ASE syntax

```
CREATE PROCEDURE GET_EMP_REC @Emp_Number int, @Emp_Name varchar(20) OUTPUT,
@Emp_Hiredate datetime OUTPUT AS
SELECT @Emp_Name = ename, @Emp_Hiredate = hiredate
FROM Employee
WHERE empno = @Emp_Number
RETURN 0
```

Related Information

Creating stored procedures in a database [page 1698]

3.6.5.4 Importing metadata for stored procedures

Use the Import by Name command to import stored procedures into Data Services. You must know the name of the stored procedure, stored function, or package you want to import. Use the name of the stored procedure, the package name only, or, if you are using a Microsoft SQL Server or SAP Sybase ASE database, use the name of the stored procedure and the group number, such as `orderproc;2`.

Stored procedures are created and structured very much like functions. Imported stored procedures appear on the Datastores tab of the object library. Each is listed in the Functions category of the database datastore from which the procedure was imported. It is displayed with the following syntax:

```
<datastore>.<owner>.<proc_name>.
```

If a stored procedure was defined within an Oracle package, the name of the procedure is displayed as

```
<datastore>.<owner>.<package>.<proc_name>.
```

To view the signature of the procedure, right-click the imported stored procedure and select Open (or simply double-click the procedure).

- A stored procedure has a signature that consists of its defined parameters including two output parameters added by Data Services during import, `AL_SP_RETCODE` and `AL_SP_ERRMSG`.
- If the database you are using provides a return code for a stored procedure, it is displayed in the signature as `AL_SP_RETURN`.
- Each parameter has a name, a data type, and a mode (`IN`, `INOUT`, or `OUT`). The mode is indicated by icons shown before the parameter name.
- Oracle stored functions have a signature that consists of parameters and a return data type. When you call a function, the return value is named `AL_SP_RETURN`.

> i Note
>
> If you change the signature of a stored procedure, for example you add a new parameter, re-import the metadata for the procedure from the database.

Related Information

Checking execution status [page 1708]

Designer Guide: Datastores, Ways of importing metadata [page 221]

3.6.5.5 Structure of a stored procedure

Imported information for a stored procedure includes:

- Name and owner
- Return data type if the stored procedure has a return value or code
- Parameters

For each parameter, Data Services imports the following information:

- Name
- Mode (`IN`, `OUT`, or `INOUT`)
- data type (length, precision and scale if applicable)

  > i Note
  >
  > In an Oracle stored procedure declaration, it is illegal to constrain char and varchar2 parameters with a length, or to constrain number parameters with a precision and/or scale. These constraints are taken from the procedure arguments. Therefore, the length of a char or varchar and the precision and scale of a number data type parameter are not stored in the data dictionary.

  However, Data Services does not allow the length of a varchar data type or the precision of a number data type to be 0. (Data Services converts char data types in Oracle to varchar.) When importing a stored procedure, Data Services fills in the default length for a varchar or the precision and scale for a number parameter. For varchar, Data Services sets the length to 4000. For number, Data Services sets precision and scale to the values set in the Advanced section of the Create New Datastore window. At runtime, the length of varchar and the precision and scale of number come from the procedure arguments.

- Position within the parameter list
- Nullable (whether a default value exists)

Data Services omits a parameter if any of the following data type conditions is true:

- Is not a Data Services supported data type or is `long`
- Data type is missing in the database dictionary
- Is a composite data type (e.g., table, record, or cursor)

After Data Services imports a stored procedure, the Designer reports warnings about any omitted parameters.

> Note
>
> Even if a stored procedure has an omitted parameter, the procedure is callable if the parameter has a default value defined. If the omitted parameter does not have a default value, the procedure is not callable and Data Services throws a runtime error.

If a stored procedure has a return value, its result type (the data type of the result value) must be a Data Services supported database server data type. Otherwise, the procedure cannot be imported and used within Data Services applications.

For each stored procedure, Data Services automatically adds two optional output parameters, `AL_SP_RETCODE` and `AL_SP_ERRMSG`. These parameters allow you to check the execution status of the stored procedure.

Related Information

Checking execution status [page 1708]
Data Types [page 1027]

3.6.5.6 Calling stored procedures

This section discusses calling stored procedures.

3.6.5.6.1 In general

You can call a stored procedure just as you call custom functions you create in Data Services.

- When you call a stored procedure inside a script, conditional or custom function, Data Services evaluates the call as an independent SQL statement.
- When you call a stored procedure in a query's column mapping or WHERE clause, (Oracle only) Data Services can combine the SQL for the stored procedure with the SQL for the query.
- You can also call a stored procedure in a SQL transform. You must use proper syntax. Data Services does not validate SQL in a SQL transform.

- A stored procedure can be used within an expression. For example:

```
ora_ds.acta_user.get_emp_sal (EMPLOYEE.NAME) + 10
ora_ds.acta_user.get_emp_sal (EMPLOYEE.NAME) < 100
```

Stored procedures can require input values for some parameters but not others. You must determine the requirements of the stored procedure and prepare the appropriate inputs. In addition, Data Services supports the following:

- Input parameters can be constants or expressions
- If an input parameter has a default value, you can omit the parameter in the call by leaving the parameter unspecified in the function wizard. The database will use the parameter's default value when evaluating the stored procedure call.
- All `OUT` or `INOUT` parameters must be specified except the two Data Services adds at import AL_SP_RETCODE and AL_SP_ERRMSG.
- All `OUT` or `INOUT` parameters must be variables (except in a query output schema). A variable must be declared before use.
- If a stored procedure returns a value (such as an Oracle function or a MS SQL Server or SAP Sybase ASE stored procedure), you can use the procedure in an expression.
- You can call a stored procedure as a step inside: an output schema of a query, script, or custom function. For example:

```
As an Oracle user you create two Oracle stored procedures: GET_EMP_SALARY and
SET_EMP_SALARY
CREATE OR REPLACE FUNCTION GET_EMP_SALARY(Emp_Name IN VARCHAR)
RETURN NUMBER AS
Return_value NUMBER;
BEGIN
SELECT sal
INTO Return_value
FROM Employee
WHERE ename = Emp_Name;
RETURN Return_value;
END;
CREATE OR REPLACE PROCEDURE SET_EMP_SALARY(Emp_Name IN VARCHAR,Emp_Sal IN
NUMBER) AS
BEGIN
UPDATE Employee
SET sal = Emp_Sal
WHERE ename = Emp_Name;
END;
```

After importing the stored procedures into an Oracle datastore, you can use the stored procedures in a script. For example:

```
$new_sal = ora_ds.bodi_user.get_emp_salary('MARTIN') + 100;
ora_ds.bodi_user.set_emp_salary('MARTIN', $new_sal);
```

Related Information

From queries [page 1706]

To include a stored procedure in a query output schema [page 1707]

3.6.5.6.2 From queries

You can call stored procedures from Query transforms.

Mapping and WHERE clauses

In a query's column mapping or WHERE clause, you can use a stored procedure in an expression, subject to the same constraints as other expressions. However, when stored procedures are used in a query's column mapping or WHERE clause, SAP Data Services can combine the SQL statement for the stored procedure call with the SQL statement for the query and submit one request to the database.

For the software to combine the SQL statements, the stored procedure must meet the following conditions:

- It must be from an Oracle database.
- It must be a stored function.
- All its parameters must be IN parameters; none can be an OUT or INOUT parameter.
- All parameter values must be specified. You cannot leave a parameter unspecified, such as to use the default value.
- All parameter values must be constants or expressions with only constants or database table column names; none can have functions or operations.
- The data types of the procedure's parameters and the return type (the data type of its result value) must be supported data types. The software must not omit any of the parameters when importing the procedure.
- Data Definition Language (DDL) operations like creating tables, views etc. and transaction control statements such as `COMMIT`/`ROLLBACK` cannot be performed inside the stored procedure.

In addition, when importing the function, you must select the Callable from SQL expression check box on the Import By Name window.

The software will push down a stored procedure call to Oracle if the first five conditions are met and the *Callable from SQL expression* check box is selected. If a stored procedure call contains any DDL statements, it ends the current transaction with `COMMIT` or `ROLLBACK`, or it issues any `ALTER SESSION` or `ALTER SYSTEM` commands. Therefore, you should deselect the *Callable from SQL expression* option on the Import By Name window to ensure that the function call will not be pushed down to database.

When a stored procedure call cannot be pushed down to database, it is submitted to database as a separate SQL statement.

Query output schemas

When a stored procedure can provide at least one return value or has any `INOUT` or `OUT` parameters other than AL_SP_RETCODE or AL_SP_ERRMSG, you can call the procedure by including it as an object in the output schema of a query.

When using a stored procedure in a query output schema, you must map `INOUT` or `OUT` parameters and return values to a column of the output schema. In this case, you cannot map these parameters to variables.

Related Information

In general [page 1704]

Importing metadata for stored procedures [page 1702]

3.6.5.6.2.1 To include a stored procedure in a query output schema

1. In a data flow diagram, click a query name to open the query editor.
2. In the query editor, right-click the query name in the output schema and select *New Function Call*.

   Alternately, you can right-click an existing stored procedure object in the output schema area of the query editor and select *Modify Function Call*.

   SAP Data Services provides a function wizard to help you include input and output parameters and the return value for a procedure call. The imported stored procedures are listed for each datastore on the first page of the function wizard.
3. Select a function category. Notice that the function names change depending on the category you select.
4. Under Function name, select a stored procedure, then click *Next*.
5. In the *Define Input Parameter(s)* window, enter the input parameters for the stored procedure, then click *Next*.
6. In the *Select Output Parameter* window, select the procedure's output parameters that you want to map to the query output.

   In the *All output parameters* box, select the parameters you want to map, and click the arrow key to move them into the *Selected output parameters* box.

   You can map more than one INOUT or OUT parameter and the return value as output columns from a single procedure call.
7. Click *Finish*.

   The software adds those parameters to the query's output schema. In this example, the output schema of the Query transform would have two columns, EMP_NAME and EMP_HIREDATE.

3.6.5.6.3 Without the function wizard

You can enter an existing stored procedure call on the *Mapping* tab or the *Where* tab in a query editor.

When entering a stored procedure call without using the function wizard:

- Match the parameter values you enter to the signature of the stored procedure in Data Services. Values must be specified in the same order that the parameters are defined.
- Use `AL_UNSPECIFIED_PARAM` in place of a missing parameter, such as when you want to use the default value for an `IN` parameter.
- Use the correct naming convention for the stored procedure:

  - <datastore>.<owner>.<proc_name>
  - <datastore>.<owner>.<package>.<proc_name>.

  If the name is case-sensitive in the database (and not all uppercase), enter the name as it appears in the database and use double quotation marks (") around the name to preserve the case.

The two output parameters added by Data Services during import, AL_SP_RETCODE and AL_SP_ERRMSG, are optional. You do not need to provide arguments for these two parameters if you are not interested in the values stored in them.

Related Information

Checking execution status [page 1708]

3.6.5.7 Checking execution status

To allow you to monitor the execution status of a stored procedure, Data Services adds two parameters to the signature of each stored procedure when importing the procedure:

- AL_SP_RETCODE, varchar(256). There are three possible values:
  - ACTA_SP_OK — The stored procedure call succeeded.
  - ACTA_SP_DB_CONN_EXCEPTION — The database connection cannot be created because of a connection error, invalid user, password, or host name.
  - ACTA_SP_CALL_ERROR — The connection completes but the call fails in the database. Details for the cause of the error are available in AL_SP_ERRMSG.
- AL_SP_ERRMSG, varchar(1024)

Data Services adds these two parameters to the end of the signature, following any user-defined parameters.

With these parameters, there are two techniques you can use to handle errors if a stored procedure fails during execution:

- Throw an error that stops the job immediately. An error message is logged in the errorlog.txt file.
  To use this method, do not map as output parameters the two parameters Data Services adds to imported stored procedures.
- Save the Oracle error messages into the AL_SP_ERRMSG parameter. This method allows the query, script, conditional, or custom function that contains the stored procedure call to continue its execution.
  To use this method, define parameters for AL_SP_RETCODE and AL_SP_ERRMSG. Use the execution status stored in AL_SP_RETCODE to control your application logic. For example, if you store the value of AL_SP_ERRMSG in a variable, you can use the print function to print the variable to the job log. With this method, the error message is not logged in the errorlog.txt file.

3.7 Scripting Language

This section describes the SAP Data Services scripting language. You can use the language to write scripts and custom functions (also known as user-script functions). Further, you can use scripting language to write expressions such as complex column mapping expressions and WHERE clause conditions.

The software uses this language internally to save the objects you create into repository tables. You can export saved objects to a file (.atl) and later import the file to another repository. This technique is commonly used when migrating to a new environment or when backing up a repository before initiating an upgrade.

3.7.1 To use the scripting language

1. Open an editor for an object in the Designer workspace such as a script, conditional, or transform.
2. Inside any of these editors, use the smart editor to enter a script or expression using the language syntax described in this section.

Related Information

Language syntax [page 1709]
Sample scripts [page 1720]

3.7.2 Language syntax

With the Data Services scripting language, you can assign values to variables, call functions, and use standard string and mathematical operators. The syntax used in the Data Services scripting language can be used in an expression as well as in a script.

3.7.2.1 Syntax for statements in scripts

Jobs and work flows can use scripts to define detailed steps in the flow of logic. A script can run functions and assign values to variables, which can then be passed to other steps in the flow.

Statements in a script object or custom function must end with a semicolon (`;`). Comment lines must start with a # character.

Related Information

Sample scripts [page 1720]

3.7.2.2 Syntax for column and table references in expressions

Expressions are a combination of constants, operators, functions, and variables that evaluate to a value of a given data type. Expressions can be used inside script statements or added to data flow objects. Because expressions can be used inside data flow objects, they often contain column names.

The Data Services Scripting Language recognizes column and table names without special syntax. For example, you can indicate the `start_date` column as the input to a function in the *Mapping* tab of a query as follows:

```
to_char(start_date, 'dd.mm.yyyy')
```

The column `start_date` must be in the input schema of the query.

If there is more than one column with the same name in the input schema of a query, indicate which column is included in an expression by qualifying the column name with the table name. For example, indicate the column `start_date` in the table `status` as follows:

```
status.start_date
```

Note that column and table names as part of SQL strings may require special syntax based on the DBMS that the SQL is evaluated by. For example, select all rows from the `LAST_NAME` column of the `CUSTOMER` table as follows:

```
sql('oracle_ds','SELECT CUSTOMER.LAST_NAME
FROM CUSTOMER')
```

Because the column name is all upper case in Oracle, no special syntax is required. However, select all rows from the `start_date` column in the Oracle table `Status` as follows:

```
sql('oracle_ds','SELECT "Status"."start_date"
FROM "Status"')
```

The table name and column are defined in Oracle to have a special case. To preserve this case, enclose the identifiers in double quotation marks.

3.7.2.3 Strings

This section contains the following information about strings:

- Quotation marks
- Escape characters
- Trailing blanks

Related Information

Quotation marks [page 1711]
Escape characters [page 1711]
Trailing blanks [page 1712]
NULL values and empty strings [page 1715]

3.7.2.3.1 Quotation marks

The type of quotation marks to use in strings depends on whether you are using identifiers or constants

Identifier	Name of an object (for example, table, column, data flow, or function).
Constant	A fixed value used in computation. There are two types of constants: ● String constants (for example, 'Hello World' or '1995.01.23') ● Numeric constants (for example, 2.14)

Identifiers need quotation marks if they contain special (non-alphanumeric) characters. For example, you need a double quote for " `compute large numbers` " because it contains blanks.

Use single quotes for string constants.

3.7.2.3.2 Escape characters

If a constant contains a single quote (`'`) or backslash (`\`) or another special character used by the Data Services scripting language, then those characters need to be escaped. For example, the following characters must be preceded with an escape character to be evaluated properly in a string. Data Services uses the backslash (\) as the escape character.

Character	Example
`single quote (')`	`'World\'s Books'`
`backslash (\)`	`'C:\\temp'`

3.7.2.3.3 Trailing blanks

Data Services does not strip trailing blanks from strings that are used in scripts or custom functions. To remove trailing blanks, use the rtrim or rtrim_blank function.

3.7.2.4 Variables

Variable names must be preceded by a dollar sign ($).

- Local variables used in a script or expression must be defined in the job or work flow that calls the script using the *Variables and Parameters* window.
- Global variables used in a script or expression must be defined at the job level using the Variables and Parameters window.
- The return value must be passed outside the function using the following statement:

  ```
  RETURN(<expression>)
  ```

  where expression defines the value to be returned.
- Local variables used in a custom function must be defined using the smart editor.
- Existing variables and parameters displayed in the smart editor are filtered by the context from which the smart editor is opened.

Related Information

Designer Guide: Variables and Parameters [page 398]

To create a custom function [page 1695]

3.7.2.5 Variable interpolation

You can embed expressions within constant strings, and Data Services will evaluate the variables and substitute the value into the string—the concatenation operator (||) is not required.

Here is an example using the concatenation operator:

```
$st_date=sql('warehouse_ds','SELECT max(start_timestamp)
    FROM dw_process_log
    WHERE table_name='\"'||$Table_Name||'\"
print('The value of the start date
    is:'||sysdate()+5);
```

The statement can be simplified to the following:

```
$st_date=sql('warehouse_ds','SELECT max(start_timestamp)
    FROM dw_process_log
    WHERE table_name={$Table_Name}'
```

```
print('The value of the start date
    is:[sysdate()+5]');
```

Curly braces ({}) and square brackets ([]) enclose the embedded expressions:

- The square brackets ([]) indicate that the value of the expression should be substituted.
- The curly braces ({}) indicate that the value of the expression should be quoted with single quotation marks.

Strings including curly braces or square brackets cause a processing error. You can avoid the error by preceding the braces or brackets with a backslash (\).

3.7.2.6 Functions and stored procedures

A script, expression or custom function can call most built-in or custom functions. A function cannot call itself or call another function that would lead to a recursive call. For example, function A cannot call function A, nor can function A call function B if function B calls function A.

You can also use functions and stored procedures imported from a DBMS. Use the exact case for names of imported functions and stored procedures

- Built-in functions are listed in the function wizard and smart editor grouped by category.
- You can find the list of Custom functions in both function wizard and the smart editor.
- Imported functions and stored procedures are listed in the function wizard and smart editorunder the datastore used to import them.

Related Information

To create a custom function [page 1695]

Importing metadata for stored procedures [page 1702]

3.7.2.7 Operators

The operators you can use in scripts and expressions are listed in the following table, in order of precedence. Note that when operations are pushed to a DBMS to perform, the precedence is determined by the rules of the DBMS.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
=	Assignment, comparison
<	Comparison, less than

Operator	Description
`<=`	Comparison, less than or equal to
`>`	Comparison, greater than
`>=`	Comparison, greater than or equal to
`!=`	Comparison, not equal to
`||`	Concatenate
`%`	Return the remainder when one number is divided by another
`AND`	Logical AND
`OR`	Logical OR
`NOT`	Logical NOT
`IS NULL`	Comparison, is a NULL value
`IS NOT NULL`	Comparison, is not a NULL value

When you use the smart editor to add operators, you can type in operators or use the built-in key pad. Access the key pad from the smart editor's tool bar.

You can use a comparison in the following ways:

- In a script or script function as a condition; for example:

  ```
  if ($x IN (1,2,3)), while ($x IN (1,2,3)) and ifthenelse()
  ```
- In a data flow such as in a WHERE clause ifthenelse() function, case transform, etc.
- As a condition of the IF block, WHILE block or TRY CATCH block

The following examples illustrate valid comparison expression syntax:

```
expression = expression
expression != expression
expression < expression
expression > expression
expression <= expression
expression >= expression
expression IS NULL
expression IS NOT NULL
expression IN (expression list)
expression IN domain
expression LIKE constant
expression NOT LIKE constant
```

`NOT` (any of the above comparisons); for example `NOT ($x IN (1,2,3))`

```
comparison OR comparison
comparison AND comparison
```

Note that the following syntax is not valid:

```
$x NOT IN (1,2,3)
EXIST or NOT EXIST
```

3.7.2.8 NULL values

Indicate NULL values using the keyword NULL. For example, you can check whether a column (COLX) is null or not:

```
COLX IS NULL
COLX IS NOT NULL
```

Data Services does not check for NULL values in data columns. Use the function `nvl` to remove NULL values.

Related Information

nvl [page 1636]

3.7.2.8.1 NULL values and empty strings

Data Services uses the following two rules with empty strings:

- When you assign an empty string to a variable of type varchar, Data Services treats the value of the variable as a zero-length string. An error results if you assign an empty string to a variable that is not a varchar. To assign a `NULL` value to a variable of any type, use the `NULL` constant.
- As a constant (`''`), Data Services treats the empty string as a varchar value of zero length. Use the `NULL` constant for the null value.

> Note
>
> Oracle does not distinguish an empty string from a `NULL` value. When you insert an empty string or a `NULL` value into a varchar column, Oracle treats both the empty string and `NULL` value as `NULL` values. Therefore, Data Services treats the value as a `NULL` value.

Data Services uses the following three rules with `NULL`s and empty strings in conditionals:

- The Equals (=) and Not Equal to (<>) comparison operators against a null value always evaluates to `FALSE`. This `FALSE` result includes comparing a variable that has a value of `NULL` against a `NULL` constant.
  The following table shows the comparison results for the variable assignments `$var1 = NULL` and `$var2 = NULL`:

Condition	Translates to	Returns
`If (NULL = NULL)`	`NULL` is equal to `NULL`	`FALSE`
`If (NULL != NULL)`	`NULL` is not equal to `NULL`	`FALSE`
`If (NULL = '')`	`NULL` is equal to empty string	`FALSE`
`If (NULL != '')`	`NULL` is not equal to empty string	`FALSE`
`If ('bbb' = NULL)`	`bbb` is equal to `NULL`	`FALSE`

Condition	Translates to	Returns
`If ('bbb' != NULL)`	`bbb` is not equal to `NULL`	`FALSE`
`If ('bbb' = '' )`	`bbb` is equal to empty string	`FALSE`
`If ('bbb' != '')`	`bbb` is not equal to empty string	`TRUE`
`If ($var1 = NULL )`	`NULL` is equal to `NULL`	`FALSE`
`If ($var != NULL)`	`NULL` is not equal to `NULL`	`FALSE`
`If ($var1 = '' )`	`NULL` is equal to empty string	`FALSE`
`If ($var != '')`	`NULL` is not equal to empty string	`FALSE`
`If ($var1 = $var2 )`	`NULL` is equal to `NULL`	`FALSE`
`If ($var != $var2)`	`NULL` is not equal to `NULL`	`FALSE`

The following table shows the comparison results for the variable assignments `$var1 = ''` and `$var2 = ''`:

Condition	Translates to	Returns
`If ($var1 = NULL)`	`Empty string` is equal to `NULL`	`FALSE`
`If ($var != NULL)`	`Empty string` is not equal to `NULL`	`FALSE`
`If ($var1 = '')`	`Empty string` is equal to empty string	`TRUE`
`If ($var != '')`	`Empty string` is not equal to empty string	`FALSE`
`If ($var1 = $var2)`	`Empty string` is equal to `Empty string`	`TRUE`
`If ($var != $var2)`	`Empty string` is not equal to `Empty string`	`FALSE`

- Use the IS NULL and IS NOT NULL operators to test the presence of null values. For example, assuming a variable is assigned: `$var1 = NULL;`

Condition	Translates to	Returns
`If ('bbb' IS NULL`	bbb is `NULL`	`FALSE`
`If ('bbb' IS NOT NULL)`	bbb is not `NULL`	`TRUE`
`If ('' IS NULL`	Empty string is `NULL`	`FALSE`
`If ('' IS NOT NULL)`	Empty string is not `NULL`	`TRUE`
`If ($var1 IS NULL)`	`NULL` is `NULL`	`TRUE`
`If ($var IS NOT NULL)`	`NULL` is not `NULL`	`FALSE`

- When comparing two variables, always test for `NULL`. In this scenario, you are not testing a variable with a value of `NULL` against a `NULL` constant (as in the first rule). Either test each variable and branch accordingly or test in the conditional as shown in the second row of the following table.

Condition	Recommendation
`if($var1 = $var2)`	Do not compare without explicitly testing for `NULL`s. Using this logic is not recommended because any relational comparison to a `NULL` value returns `FALSE`.
`if ( (($var1 IS NULL) AND ($var2 IS NULL)) OR ($var1 = $var2))`	Will execute the `TRUE` branch if both `$var1` and `$var2` are `NULL`, or if neither are `NULL` but are equal to each other.

3.7.2.9 Debugging and Validation

If you are using the smart editor to create a script or a custom function, select the *Validate* icon in the tool bar (or right-click and select *Validate*). Errors are listed in the smart editor window under the text box. Double-click each error to see where it occurred in your script.

Otherwise:

- Select the *Debug* > *Validate* > *Current View* command to check the syntax of expressions used in the definition of the current object. Errors are displayed in the Output window.
- Select the *Debug* > *Validate* > *All Objects in View* command to check the syntax of expressions used in the current object definition and the definitions of all objects called by the current object. Errors are displayed in the Output window.

By default, Data Services lists ten errors before it aborts parsing. By reporting ten errors for each validation, Data Services allows you to shorten the edit-validate-test cycle. You can repair several errors in each iteration of the cycle.

If you want to change this default setting, select *Tools* > *Options* > *Job Server* > *General* and set the options as follows:

Section:	Enter "Parser".
Key:	Enter "NumErrors".
Value:	Enter a positive number.

For each error, Data Services provides a description of the problem.

Data Services also provides three menu options for errors. Right-click an error to use these menu options:

View: The *View* option allows you to view the error text in a smaller, separate window for easy reading.

Go To Error: The *Go to Error* option places an error icon near the line that caused the error. You can also double-click the error to place this icon. This option not available for all errors, however.

Copy: The *Copy* option copies the error text on to the clipboard so you can place it elsewhere.

When you execute a job, you may still encounter errors in an expression that are returned from the operating system or the DBMS. When executing a job, Data Services sends as many operations as possible to the DBMS to evaluate. The DBMS may evaluate part of the expression and return a value that contributes to evaluating the rest of the expression.

3.7.2.10 Keywords

The following sections describe the keywords in the scripting language.

Keywords are listed in the selection list of the smart editor filtered by the context from which the smart editor is opened.

Related Information

About Reserved Words [page 1847]

3.7.2.10.1 BEGIN

Indicates the beginning of the code that becomes the function, script, or other construct. BEGIN and END statements are added automatically to function, transform, and script definitions.

3.7.2.10.2 CATCH

Indicates a catch for a try/catch block. If an error occurs while executing any of the statements between the TRY and the CATCH statements, Data Services executes the statements defined by the CATCH. Use the CATCH keyword as shown in the following script, or use `CATCH(ALL)`.

```
BEGIN
TRY
    BEGIN
        <script_step>;
        <script_step>;
    END
CATCH (<exception_number>)
    BEGIN
        <catch_step>;
        <catch_step>;
    END
CATCH (<exception_number>)
    BEGIN
        <catch_step>;
        <catch_step>;
    END
END
```

3.7.2.10.3 ELSE

Defines the second branch for an IF statement. If no ELSE follows the IF, no action is taken if the condition is not met.

3.7.2.10.4 END

Indicates the end of the code that becomes the function, script, or other construct. BEGIN and END statements are added automatically to function, transform, and script definitions.

3.7.2.10.5 IF

Indicates a conditional step in the code. An IF statement can be constructed with or without an ELSE step. Use the IF keyword as follows:

```
IF (<condition>) <script_step>; ELSE <script_step>;
```

or

```
IF (<condition>) <script_step>;
```

where `<condition>` is an expression that evaluates to True or False. `<script_step>` indicates the set of instructions to execute based on the value of `<condition>` . If `<script_step>` contains more than one statement, enclose these statements in BEGIN and END statements.

Related Information

RepeatString function [page 1720]

3.7.2.10.6 RETURN

Indicates the value to be returned by a function. Use the RETURN keyword as follows:

```
RETURN (<expression>);
```

where `<expression>` defines the value to be returned.

3.7.2.10.7 TRY

Indicates the start of a try/catch block.

Related Information

CATCH [page 1718]

3.7.2.10.8 WHILE

Defines a set of statements to execute until a condition evaluates to FALSE. Use the WHILE keyword as follows:

```
WHILE (<condition>) <script_step>;
```

where **<condition>** is an expression that evaluates to True or False. **<script_step>** indicates the set of instructions to execute based on the value of **<condition>** . If **<script_step>** contains more than one statement, enclose each statement in BEGIN and END statements.

3.7.3 Sample scripts

The following examples show how a script would be formatted for the Square and Repeat String functions.

3.7.3.1 Square function

This function accepts an integer and returns the square of the input value. To define this script, supply the following values to Data Services:

Description	Value	Arguments
Function name	Square	
Return value	Return	int
Input value	input_int	int, input

The text of the function script is as follows:

```
RETURN($input_int * $input_int);
```

3.7.3.2 RepeatString function

This function accepts a one-character string and an integer indicating the number of times to repeat the input character. It outputs the created string or, if the input character or repeat number is NULL, it outputs a NULL value. It uses a variable to keep track of the string components as it is being generated.

To define this script, supply the following values to Data Services:

Description	Value	Arguments
Function name	RepeatString	
Return value	Return	VARCHAR (255),allow NULL
Input string	InputCharacter	VARCHAR (1), input, allow NULL
Input integer	RepeatNumber	INT, input, allow NULL
Internal variable	Partial	VARCHAR (255), allow NULL

The text of the function is as follows:

```
IF ($InputCharacter = NULL)
    BEGIN
    $Partial = NULL;
    RETURN NULL;
    END
IF ($RepeatNumber = NULL)
    RETURN NULL;
IF ($RepeatNumber > 255)
    $RepeatNumber = 255;
IF ($RepeatNumber < 0)
    $RepeatNumber = 0;
    BEGIN
    WHILE ($RepeatNumber != 0)
        BEGIN
        $Partial = ($Partial || $InputCharacter);
        $RepeatNumber = ($RepeatNumber - 1);
        END
    RETURN $Partial;
    END
```

3.8 Metadata in Repository Tables and Views

Data Services provides full access to the repository metadata used by the applications you create. To access this metadata for application analysis, either manually enter SQL SELECT statements or open the metadata reporting tool.

This section describes the tables and views in the repository that are useful for metadata analysis. This section also provides sample SQL SELECT statements for creating reports using a SQL editor.

For more information, see the *Management Console Guide*.

3.8.1 Auditing metadata

This set of tables in the Data Services repository stores the statistics that the audit feature collects.

- *AL_AUDIT* [page 1722]
- *AL_AUDIT_INFO* [page 1722]

3.8.1.1 AL_AUDIT

This table contains audit information about each data flow execution. The column OBJECT_KEY uniquely identifies a data flow execution.

Column Name	Data Type	Description
OBJECT_KEY	INTEGER	Identifies the audit event.
HISTORY_KEY	INTEGER	Refers to the OBJECT_KEY column in the AL_HISTORY table. Use this referential relationship to obtain history information about operational statistics for the data flow.
DF_LANG_KEY	INTEGER	Refers to the OBJECT_KEY column in the AL_LANG table. Use this referential relationship to obtain the definition for the data flow.
STATUS	INTEGER	Audit status can be one of the following values: ● 0 — Not audited ● 1 — Audit rule succeeded ● 2 — Audit information collected. This status occurs when you define audit labels to collect statistics but do not define audit rules. ● 3 — Audit rule failed
RULEINFO	VARCHAR(255)	The audit rule that failed and the values of the left-hand-side (LHS) and right-hand-side (RHS) of the Boolean expression.

Example

The following query returns the names of data flows that failed an audit and the audit rules that failed.

```
SELECT L.NAME, RULEINFO
    FROM AL_AUDIT A, AL_LANG L
    WHERE A.STATUS = 3
        AND A.DF_LANG_KEY = L.OBJECT_KEY
```

3.8.1.2 AL_AUDIT_INFO

This table contains information about the audit statistics.

Column Name	Data Type	Description
AUDIT_KEY	INTEGER	Refers to the OBJECT_KEY column in the AL_AUDIT table. Use this referential relationship to obtain audit information for the data flow.
LABEL	VARCHAR(255)	Refers to the OBJECT_KEY column in the AL_HISTORY table. Use this referential relationship to obtain history information for the data flow.
VALUE	VARCHAR(255)	Value of the label. This value can be one of the following: ● Number of rows in a table or whole row set

Column Name	Data Type	Description
		- Sum of the values in a column - Average of the values in a column - Checksum of the values in a column

Example

The following query returns the values of the labels of audit points that failed.

```
SELECT L.NAME DF_NAME, B.LABEL, B.VALUE AUDIT_VALUE
    FROM AL_AUDIT A, AL_AUDIT_INFO B, AL_LANG L
    WHERE A.OBJECT_KEY = B.AUDIT_KEY
        AND A.STATUS = 3
        AND A.DF_LANG_KEY = L.OBJECT_KEY;
```

If you want to see values of audit labels for a specific data flow, use a query similar to the following:

```
SELECT L.NAME DF_NAME, B.LABEL, B.VALUE AUDIT_VALUE
    FROM AL_AUDIT A, AL_AUDIT_INFO B, AL_LANG L
        WHERE A.OBJECT_KEY = B.AUDIT_KEY
            AND A.STATUS = 3
            AND A.DF_LANG_KEY = L.OBJECT_KEY
            AND L.NAME = 'Validation_DF';
```

3.8.2 Imported metadata

This set of tables (prefixed by AL_) and views (prefixed by ALVW_) capture information about the metadata imported into SAP Data Services from external databases and applications (such as Oracle, PeopleSoft, and SAP Applications).

- *AL_INDEX* [page 1723]
- *AL_PCOLUMN* [page 1724]
- *AL_PKEY* [page 1724]
- *ALVW_COLUMNATTR* [page 1725]
- *ALVW_COLUMNINFO* [page 1726]
- *ALVW_FKREL* [page 1727]
- *ALVW_MAPPING* [page 1727]
- *ALVW_TABLEATTR* [page 1733]
- *ALVW_TABLEINFO* [page 1734]

3.8.2.1 AL_INDEX

This table contains index information for tables.

Column Name	Description
TABLEKEY	The table ID associated with the index. The TABLEKEY relates to table information in ALVW_TABLEINFO.
NAME	The index name.
COLNAME	The name of the table's index column.
COLPOSITION	Position of the column in the index.

Example

The following query returns the index information associated with a table EMPLOYEE in datastore ORA_DS:

```
SELECT NAME, COLNAME, COLPOSITION
    FROM AL_INDEX, ALVW_TABLEINFO
    WHERE AL_INDEX.TABLEKEY =
            ALVW_TABLEINFO.TABLE_ID
        AND TABLE_NAME = 'EMPLOYEE'
        AND DATASTORE = 'ORA_DS'
```

3.8.2.2 AL_PCOLUMN

This table contains partition information for tables.

Column Name	Description
TABLEKEY	The partitioned table ID. The TABLEKEY relates to table information in ALVW_TABLEINFO.
COLNAME	The name of the table's partition column.
COLPOSITION	Position of the column in the partition.

3.8.2.3 AL_PKEY

This table contains primary key information for tables.

Column Name	Description
TABLEKEY	The table ID associated with a primary key. The TABLEKEY relates to table information in ALVW_TABLEINFO.
COLNAME	Name of the table's primary key column.

Column Name	Description
COLPOSITION	Position of the primary key column.

Example

The following query returns the primary key associated with a table EMPLOYEE in datastore ORA_DS:

```
SELECT COLNAME, COLPOSITION
    FROM AL_PKEY, ALVW_TABLEINFO
    WHERE AL_PKEY.TABLEKEY =
            ALVW_TABLEINFO.TABLE_ID
        AND TABLE_NAME = 'EMPLOYEE'
        AND DATASTORE = 'ORA_DS'
```

3.8.2.4 ALVW_COLUMNATTR

This view contains information about column attributes.

Column Name	Data type	Description
TABLE_NAME	Varchar(256)	Name of the table.
TABLE_OWNER	Varchar(256)	Owner of the table.
DATASTORE	Varchar(256)	The datastore to which this table belongs.
COLUMN_NAME	Varchar(256)	Name of the column.
COLUMN_ATTR	Varchar(64)	Name of the attribute (property of this table). The name of the attribute will be the same as seen when viewing the attributes of a table in the Designer's interface. Examples: Description, Business_Name, Date_last_loaded, Date_created, Total_Number_Of_Rows_Processed.
COLUMN_ATTR_VALUE	Varchar(255)	Value of this attribute.

Example

The following query returns all columns and their descriptions for table EMP in datastore HR:

```
SELECT COLUMN_NAME, COLUMN_ATTR_VALUE
    FROM ALVW_COLUMNATTR
    WHERE TABLE_NAME = 'EMP'
```

```
        AND COLUMN_ATTR = 'Description'
        AND DATASTORE = 'HR'
```

3.8.2.5 ALVW_COLUMNINFO

This view contains information about the columns in a table.

Column Name	Data type	Description
TABLE_NAME	Varchar(256)	Name of the table.
TABLE_OWNER	Varchar(256)	Owner of the table.
DATASTORE	Varchar(256)	The datastore to which this table belongs.
COLUMN_NAME	Varchar(256)	Name of the column.
COLUMN_DATATYPE	Varchar(20)	Data type of this column. Examples: `integer`, `datetime`, `decimal`, `real`.
COLUMN_LENGTH	Int	Length of this column in bytes.
COLUMN_PRECISION	Int	Precision of this column (applies only to columns with the data type `decimal`).
COLUMN_SCALE	Int	Scale for this column (applies only to columns with the data type `decimal`).
COLUMN_IS_NULLABLE	Int	One (`1`) indicates this column will accept NULL values, `0` indicates not.
COLUMN_ID	Int	The ID for this column within the repository. It can be used to join with other tables containing additional column information.

Example

The following query returns all columns in table `EMP` in datastore `HR`:

```
SELECT COLUMN_NAME
    FROM ALVW_COLUMNINFO
    WHERE TABLE_NAME = 'EMP'
        AND DATASTORE = 'HR'
```

3.8.2.6 ALVW_FKREL

This view contains information about the primary and foreign key relationships among tables.

Column Name	Description
TABLEKEY	The table ID.
PKCOLUMN	Primary key column in this table.
FKTABLE	The table where this column is referenced.
FKCOLUMN	The column in the 'foreign' table which is the same as the primary column.

Example

The following query returns the primary and foreign key information associated with table **<EMPLOYEE>** in datastore **<ORA_DS>**.

```
SELECT FKTABLE, FKCOLUMN, PKCOLUMN
    FROM ALVW_FKREL, ALVW_TABLEINFO
    WHERE ALVW_FKREL.TABLEKEY =
            ALVW_TABLEINFO.TABLE_ID
        AND TABLE_NAME='EMPLOYEE'
        AND DATASTORE='ORA_DS'
```

3.8.2.7 ALVW_MAPPING

The ALVW_MAPPING view joins the AL_COLMAP and the AL_COLMAP_TEXT tables. These tables contain information about target tables and columns, the sources used to populate target columns, and the transforms Data Services applies to sources before applying them to targets. Data Services uses the ALVW_MAPPING view for impact analysis in Metadata Reports.

The column mapping calculation generates the following information for target columns:

- The source column(s) from which the target column is mapped.
- The expressions used to populate target columns.

Data Services stores column mappings of nested source and target data in data flows using both the ALVW_MAPPING view and the AL_COLMAP_NAMES table.

Note

For those column names with varchar(256), the maximum length depends on the Data Services repository type. For most repository types the maximum length is 256, for MySQL the length is 64, and for MS SQL server the length is 128.

Table 176: ALVW_MAPPING view

Column Name	Data type	Description
DF_NAME	varchar(256)	Data flow that populates the target table.
TRG_TAB_NAME	varchar(256)	Name of the target table.
TRG_TAB_ID	int	ID for this table within the repository.
TRG_TAB_DESC	varchar(100)	Description of the target table.
TRG_OWNER	varchar(256)	Owner of the target table.
TRG_DS	varchar(256)	Datastore of the target table.
TRG_TYPE	varchar(64)	Type of target. Examples: table, BW transfer structure.
TRG_USAGE	varchar(65)	Usage of the target table. Examples: fact, dimension, lookup. Currently set to NULL.
TRG_COL_NAME	varchar(256)	Column name in the target.
TRG_COL_ID	int	ID for this column in the repository.
TRG_COL_DESC	varchar(100)	Description of this column.
SRC_TAB_NAME	varchar(256)	Name of the source table used to populate the target.
SRC_TAB_ID	int	ID for this table within the repository.
SRC_TAB_DESC	varchar(100)	Description of the source table.
SRC_OWNER	varchar(256)	Owner of the source table.
SRC_DS	varchar(256)	Datastore of the source table.
SRC_TYPE	varchar(64)	Type of source. Examples: table, file.
SRC_COL_NAME	varchar(256)	Name of the source column.
SRC_COL_ID	int	ID for this column in the repository.
SRC_COL_DESC	varchar(100)	Description of this column.
MAPPING_TYPE	varchar(65)	Types of source to target mapping. Examples: direct, computed, lookup.
MAPPING_TEXT	varchar(255)	The expression used to map the source to the target column.

Related Information

Storing nested column-mapping data [page 1731]

3.8.2.7.1 Example use case

The following query returns target tables and columns populated from the column EMPID in table EMP (in datastore HR):

```
SELECT TRG_TAB_NAME, TRG_COL_NAME
    FROM ALVW_MAPPING
    WHERE SRC_TAB_NAME = 'EMP'
        AND SRC_COL_NAME = 'EMPID'
        AND SRC_DS = 'HR'
```

3.8.2.7.2 Mapping types

The AL_COLMAP_TEXT table contains information qualifying the mapping relationships. This information, stored in the MAPPING_TYPE column, can have the following values:

Mapping Type	Description
Direct	The target column is mapped directly from a source column with no expression to transform it. For example, EMPID (employee ID) mapped directly from source to target.
Computed	There is an expression associated with the target column. For example, NAME is LAST_NAME||','|| FIRST_NAME.
Generated	There is no source column associated with the target column. For example, the target table is mapped to a constant or a function, such as sysdate, or is obtained from a transform, such as Date_Generation.
LookedUp	A lookup function is used in the expression.
Merged	Two data streams are merged to populate the target table. The two expressions mapped to the target table are separated by AND.
Not mapped	The column in the target table is not being populated by the data flow.
Unknown	Data Services is unable to identify the expression used to map the target column. This happens only under unusual error conditions.

3.8.2.7.3 How mappings are computed

When a data flow processes information, it performs potentially complex transformations on data in preparation for loading it into one or more target tables. Typical operations include:

- Reading data from the appropriate sources.
- Processing data using Query transforms or other transforms.
- Splitting the data stream and then merging it again.

Consider the following example, where two transformations operate against a value from one column of a source table.

The information is captured in AL_COLMAP_TEXT as follows:

Target column	Source column	Mapping expression
Total_value	Price	`((Price x 1.17) x 112)`

This kind of information becomes more valuable as transformation complexity increases. Consider the following example:

- Data flow DF_1 reads three columns (a, b, c) from source table S.
- Table S is connected to a Query transform Q1.
- The Query transform output has four columns (Qa, Qb, Qc, and Qd) whose mapping expressions are S.a, S.b, S.c and S.a – S.b.
- The output of Q1 is connected to Query transform Q2, which has two columns Q2y and Q2z whose expressions are Qa – Qb and Qc – Qd.
- The output of Q2 is loaded into target table T, which has two columns: T1 and T2.

The mapping expressions for target columns T1 and T2 are computed by starting from the end point (the target object) and "walked" back through the list of transforms, with columns of a transform written in terms of expressions from the previous transform.

When processing is started on data flow DF_1, it starts with column T1 of target table T.

The expression for T1 is Q2y, which in turn is A – dB, which can be written as S.a – S.b. Therefore the mapping expression is S.a – S.b and column T1 has two source columns—it is mapped from S.a and S.b. The AL_COLMAP table contains two rows for the target column to describe the two source columns.

In the case of T2, it is mapped from DC – JD, which can be written as S.c – (S.a – S.b). In this case, there are three rows for this target column in the AL_COLMAP table, one for each source column.

3.8.2.7.4 Mapping complexities

If a data flow calls another data flow and then loads a target table, the mappings are expressed in terms of the tables and columns used within the other data flow. Information is generated by "drilling down" into the other data flow to continue the mapping process.

The situation in which the Merge transform is used within a data flow is a bit more complex, because when two data streams are merged, there are two ways to populate a target table. This possibility is captured by separating the mapping expressions with the keyword `AND`. For example, a target column could be populated from `S.a AND R.a`.

Transforms like Hierarchy_Flattening and Pivot also introduce column mapping complexities.

It is also possible that some target columns are mapped by constants or expressions that do not use source columns. In this case there will be no rows in the AL_COLMAP table for the target column. The mapping expression in the AL_COLMAP_TEXT table will reflect this.

If a target column is populated with a call to the lookup function, then its source columns are both the looked up column and the key used to do the lookup.

3.8.2.7.5 Storing nested column-mapping data

SAP Data Services calculates column mappings (identifies the source column(s) and expressions in use to map the target column) for all data flows including those that use nested data.

The following objects and conditions are supported:

- XML files or messages
- IDOC files or messages
- Custom and adapter functions
- SAP Applications and PeopleSoft hierarchies
- Column mappings that perform nesting or un-nesting (target columns mapped from a nested or un-nested data set)
- Nested columns used as parameters in custom or adapter functions (including SAP Application RFC output parameters, BAPI function calls, and database stored procedures)
- Embedded data flows
- ABAP data flows
- Correlated columns
  You can map a column in a nested schema from a column in the input schema of the nested schema, or from a column in the input schema of the parent (or any ancestor) of the nested schema. If you map a column from an ancestor, the column is correlated.

Transforms support nested column-mapping data as follows:

- Query transforms process nested data and mappings are stored in repository tables and views.
- The software allows nested column mappings to pass through the Merge, Case, and Map_Operation transforms.
- Other transforms do not process nested data.

Nested (NRDM) notations that represent column names are longer than those used for a flat schema column.

- A column in a flat schema is represented by Table. Column, for example, "mats_emp.empno". Note that Table may represent a database table, a file, an XML message or file, an IDOC message or file, and so on.
- A column in a nested schema is represented by

  ```
  Table.Subschema1…SubschemaN.Column
  ```

For example, "personnel.name.given" represents a column of a nested schema which has three components. The first component is the Table. The last component is the Column. The middle components identify the nested levels in the Table.

Because the TRG_COL_NAME and SRC_COL_NAME columns in the ALVW_MAPPING view of the repository are VARCHAR(64) (MySQL only) and not big enough to store long NRDM column names, the software uses the AL_COLMAP_NAMES table to support nested data.

Table 177: AL_COLMAP_NAMES table

Column Name	Data type	Description
DF_NAME	varchar(256)	Data flow with that populates a target table.
COL_ID	varchar(65)	The software generates this value using the following format when a nested column is encountered. `"__DI_NESTED_COLNAME_n"` where n is an integer that starts from 1
COL_NAME	varchar(256)	If the software generates a COL_ID value, this column stores the original nested column name.
SEQNUM	int	The software generates this value if more than one set of 256 characters is required to store data in the COL_NAME column. For each set of 256 characters, it generates a new row and a sequence number.

The AL_COLMAP_NAMES table uses the DF_NAME, COL_ID, SEQNUM columns as primary keys. The DF_NAME and COL_ID columns are keyed to the following columns in the ALVW_MAPPINGS view.

- DF_Name is keyed to DF_Name.
- COL_ID is keyed to SRC_COL_NAME and TRG_COL_NAME

The AL_COLMAP_NAMES table also provides an internal mapping mechanism from COL_ID column to COL_NAME.

For example, if a source column BOOKS.AUTHOR.FIRST_NAME is mapped into a target column BOOK.AUTHOR_NAME (an un-nesting is probably in place), you can create a report to query the following column values in the repository:

ALVW_MAPPING view	Column value
TRG_TAB_NAME	BOOK
TRG_COL_NAME	AUTHOR_NAME
SRC_TAB_NAME	BOOKS
SRC_COL_NAME	__DI_NESTED_COLNAME_1
MAPPING_TEXT	BOOKS.AUTHOR.FIRST_NAME

AL_COLMAP_NAMES table	Column value
COL_ID	__DI_NESTED_COLNAME_1

AL_COLMAP_NAMES table	Column value
COL_NAME	AUTHOR.FIRST_NAME

The TRG_COL_NAME or SRC_COL_NAME columns in the ALVW_MAPPING view store the COL_ID, if the target or source column is nested. To get the actual column name, lookup the AL_COLMAP_NAMES table using the DF_Name, COL_ID, and COL_NAME.

Flat or un-nested target or source column names are stored using the format Column in TRG_COL_NAME and SRC_COL_NAME. For example, of the three source columns shown below, only the second one is nested:

SRC_COL_NAME
EMPNO
_DI_Nested_Names_1
ENAME

The second value is the only one for which the software generates a column ID. To find this source column's real name, create a report that looks up its COL_NAME from the AL_COLMAP_NAMES table.

3.8.2.8 ALVW_TABLEATTR

This view contains information about the attributes in a table.

Column Name	Data type	Description
TABLE_NAME	Varchar(256)	Name of the table.
TABLE_OWNER	Varchar(256)	Owner of the table. For SAP applications, the value is NULL.
DATASTORE	Varchar(256)	The datastore to which this table belongs.
TABLE_ATTR	Varchar(64)	Name of the attribute (property of this table). The name of the attribute will be the same as seen when viewing the attributes of a table in the Designer's interface. Examples: Description, Business_Name, Date_last_loaded, Date_created, Total_Number_Of_Rows_Processed.
TABLE_ATTR_VALUE	Varchar(255)	Value of this attribute.

Example

The following query returns when table EMP in datastore HR was last loaded:

```
SELECT TABLE_ATTR_VALUE
    FROM ALVW_TABLE_ATTR
    WHERE TABLE_NAME = 'EMP'
        AND TABLE_ATTR = 'Date_last_loaded'
        AND DATASTORE = 'HR'
```

3.8.2.9 ALVW_TABLEINFO

This view contains the list of tables imported into the repository.

Column Name	Data type	Description
TABLE_NAME	Varchar(256)	Name of the table.
TABLE_OWNER	Varchar(256)	Owner of the table. For SAP applications, this table is NULL.
DATASTORE	Varchar(256)	The datastore to which this table belongs.
TABLE_ID	int	Internal ID of the table within the repository.

3.8.3 Internal metadata

This set of tables and views capture information about built-in metadata objects used by Data Services and the relationships between those objects.

- *AL_LANG* [page 1734]
- *AL_LANGXMLTEXT* [page 1735]
- *AL_ATTR* [page 1736]
- *AL_USAGE* [page 1737]
- *ALVW_FUNCINFO* [page 1740]
- *ALVW_PARENT_CHILD* [page 1741]

3.8.3.1 AL_LANG

This table contains various Data Services objects. These objects are also displayed in Data Services' object library.

Column Name	Description
OBJECT_KEY	Internal ID of the object.
OBJECT_TYPE	Type of object (an integer). Query the AL_REPO-TYPE_NAMES table to find the corresponding name for this type. Examples: data flow, table, datastore.
NAME	Name of the object.
VERSION	Indicates the number of times this object has been updated.
TYPE	Subtype of the object.
OWNER	For table objects, the owner.
DATASTORE	For table objects, the datastore to which they belong.
NORMNAME	Unique name for this object in this table.

Example

The following query returns all the data flows in the repository:

```
SELECT OBJECT_KEY, NAME
    FROM AL_LANG A
    WHERE OBJECT_TYPE = 1
        AND VERSION =
            (SELECT MAX(VERSION) FROM AL_LANG B
            WHERE A.NORMNAME = B.NORMNAME)
```

3.8.3.2 AL_LANGXMLTEXT

This table contains various Data Services objects. These objects are also displayed in Data Services' object library.

Column Name	Description
OBJECT_KEY	Internal ID of the object.
DATE_MODIFIED	Date the object was last modified.
OBJECT_TYPE	Type of object (an integer). Query the AL_REPO-TYPE_NAMES table to find the corresponding name for this type. Examples: data flow, table, datastore.

Column Name	Description
OBJECT_SUBTYPE	Subtype of the object.
OBJECT_NORMNAME	Unique name for this object in this table.
SEQNUM	Used reorder XML segments.
TEXT_VALUE	Contains the XML representation of the object.

3.8.3.3 AL_ATTR

This table contains attributes of repository objects provided by Data Services. For example, a description is an object attribute.

Column Name	Description
PARENT_OBJID	The internal ID of an object.
PARENT_OBJ_TYPE	Type of the object (integer).
ATTR_NAME	The attribute name for this object.
ATTR_VALUE	Value of the attribute.

Example

The following query returns all the data flows and their descriptions in the repository:

```
SELECT OBJECT_KEY, NAME, ATTR_VALUE
    FROM AL_LANG O, AL_ATTR A
    WHERE O.OBJECT_TYPE = 1
        AND O.OBJECT_KEY = A.PARENT_OBJID
        AND ATTR_NAME = 'Description'
        AND O.VERSION =
            (SELECT MAX(VERSION) FROM AL_LANG B
            WHERE O.NORMNAME = B.NORMNAME)
```

3.8.3.4 AL_SETOPTIONS

This table includes option settings for all objects.

Column Name	Data type	Description
PARENT_OBJ_ID	numeric(38)	ID of the parent object to which this option belongs.
PARENT_OBJ_TYPE	numeric(38)	Type of the parent object. Examples: batch job or data flow.
CALL_OBJID	varchar(100)	ID of the calling object. (Future use)
CALL_OBJTYPE	varchar(255)	Type of the calling object. (Future use)
OPTION_NAME	varchar(100)	Name of the option.
OPTION_VALUE	varchar(255)	Value of the option.
OVERFLOW_KEY	numeric(38)	KEY value pointing to a row in the AL_OVERFLOW_ATTR table. When an OPTION_VALUE exceeds 255 characters, Data Services adds the remaining characters to AL_OVERFLOW_ATTR and stores the row ID as OVERFLOW_KEY in the AL_SETOPTIONS table.

3.8.3.5 AL_USAGE

This table is identical to ALVW_PARENT_CHILD except it captures the entire call hierarchy. For example, if a table is used in a data flow which is called from a work flow, then rows appear in this table that associate the work flow (parent) to the table (descendent). The Depth column is unique to this table.

Note

You need to populate this table explicitly by using the *Calculate Usage Dependencies* command.

To query this table, use the following *Example use cases* [page 1738].

Column Name	Data type	Description
PARENT_OBJ	varchar(64)	Name of the calling object.
PARENT_OBJ_TYPE	varchar(64)	Type of the object. Examples: batch job or data flow.
PARENT_OBJ_DESC	varchar(255)	The description associated with this object.
PARENT_OBJ_KEY	int(4)	Key in the AL_LANG table of the parent object.
DESCEN_OBJ	varchar(64)	Name of the descendant object.

Column Name	Data type	Description
		For transforms, the name of the output schema of the transform call (if the name of the transform is unique). If it is not unique, Data Services generates a unique numeric suffix and appends that to the given name.
DESCEN_OBJ_TYPE	varchar(64)	Type of the called object. Examples: data flow, table, function, file.
DESCEN_OBJ_DESC	varchar(255)	Description associated with the called object.
DESCEN_OBJ_USAGE	varchar(20)	Applies only to tables and files. How the child is used. Examples: source, target, lookup table.
DESCEN_OBJ_KEY	int(4)	Key in AL_LANG of the descendant object.
DESCEN_OBJ_DS	varchar(64)	Applies only to tables and files. The datastore of the child object.
DESCEN_OBJ_OWNER	varchar(256)	Owner of the child table.
DEPTH	int	Indicates the number of levels between objects in a parent/descendent relationship.

3.8.3.5.1 Example use cases

You can query the AL_USAGE table using SQL statements. For example, the following report shows a few columns and rows from AL_USAGE listing the objects that are related to the SALES_ORDER target table:

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
Build_Fact	data flow	SALES_ORDER	Table	Sales order target fact table	Target
Daily_Job	work flow	SALES_ORDER	Table	Sales order target fact table	Target
DF_NewOrders	Job	SALES_ORDER	Table	Sales order target fact table	Target
Get_IDoc	data flow	SALES_ORDER	Table	Sales order target fact table	Target
IDoc_job	data flow	SALES_ORDER	Table	Sales order target fact table	Target
initial_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target

Example

To find out which jobs load a table

The following query returns which jobs load a particular target:

```
SELECT PARENT_OBJ
    FROM AL_USAGE
    WHERE PARENT_OBJ_TYPE = 'Job'
        AND DESCEN_OBJ_TYPE = 'Table'
        AND DESCEN_OBJ_USAGE = 'Target'
        AND DESCEN_OBJ = '<targetTable>'
```

Example

To find out which objects depend on a source

The following query returns which objects would be affected if you drop a source table:

```
SELECT PARENT_OBJ, PARENT_OBJ_TYPE
    FROM AL_USAGE
    WHERE DESCEN_OBJ_TYPE = 'Table'
        AND DESCEN_OBJ = '<targetTable>'
```

Example

To produce a "where used" report for an object

The following query returns what objects call a given object. The given object in this case is the target table SALES_ORDER.

```
SELECT
            AL_USAGE.PARENT_OBJ,
            AL_USAGE.PARENT_OBJ_TYPE,
            AL_USAGE.DESCEN_OBJ,
            AL_USAGE.DESCEN_OBJ_DESC,
            AL_USAGE.DESCEN_OBJ_TYPE,
            AL_USAGE.DESCEN_OBJ_USAGE
    FROM AL_USAGE
    WHERE (AL_USAGE.DESCEN_OBJ_TYPE = 'Table'
            AND AL_USAGE.DESCEN_OBJ='SALES_ORDER')
```

The following table shows the result for an example repository:

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
Build_Fact	data flow	SALES_ORDER	Table	Sales order target fact table	Target
Daily_Job	work flow	SALES_ORDER	Table	Sales order target fact table	Target
DF_NewOrders	Job	SALES_ORDER	Table	Sales order target fact table	Target
Get_IDoc	data flow	SALES_ORDER	Table	Sales order target fact table	Target

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
IDoc_job	data flow	SALES_ORDER	Table	Sales order target fact table	Target
initial_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target

Example

To show which jobs load what targets

The following query returns all of the jobs in the repository and what targets they load:

```
SELECT
    AL_USAGE.PARENT_OBJ,
    AL_USAGE.PARENT_OBJ_TYPE,
    AL_USAGE.DESCEN_OBJ,
    AL_USAGE.DESCEN_OBJ_DESC,
    AL_USAGE.DESCEN_OBJ_TYPE,
    AL_USAGE.DESCEN_OBJ_USAGE
    FROM AL_USAGE
    WHERE (AL_USAGE.PARENT_OBJ_TYPE = 'Job'
        AND AL_USAGE.DESCEN_OBJ_TYPE = 'Table'
        AND AL_USAGE.DESCEN_OBJ_USAGE = 'Target')
```

This table shows the results for an example repository:

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
Daily_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target
IDoc_job	Job	SALES_ORDER	Table	Sales order target fact table	Target
initial_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target

3.8.3.6 ALVW_FUNCINFO

This view contains a list of functions you defined in Data Services and those you imported into its repository.

Column Name	Description
FUNC_KEY	Internal ID for the function within the repository.
FUNC_NAME	The function name.
FUNC_OWNER	Applies only to imported functions. Owner of the function.

Column Name	Description
DATASTORE	Applies only to imported functions. Datastore to which the function belongs.

3.8.3.7 ALVW_PARENT_CHILD

This view contains information about objects (parents) that contain (or call) other objects (children).

Column Name	Description
PARENT_OBJ	Name of the calling object.
PARENT_OBJ_TYPE	Type of the object. Examples: batch job, real-time job, data flow.
PARENT_OBJ_DESC	The description associated with this object.
DESCEN_OBJ	Name of the called object.
DESCEN_OBJ_TYPE	Type of the called object. Examples: data flow, table, function, file.
DESCEN_OBJ_DESC	Description associated with the called object.
DESCEN_OBJ_USAGE	Applies only to tables and files. How the child is used. Examples: source, target, lookup table.
DESCEN_OBJ_DS	Applies only to tables and files. The datastore of the child object.
DESCEN_OBJ_OWNER	Owner of the child table.

Example

You can query this table to:

- View which data flows load table EMP

```
SELECT PARENT_OBJ
    FROM ALVW_PARENT_CHILD
    WHERE DESCEN_OBJ_TYPE = 'Table'
        AND DESCEN_OBJ = 'EMP'
        AND DESCEN_OBJ_USAGE = 'Target'
```

- View the data flows called by job HR_INITIAL_LOAD

  ```
  SELECT DESCEN_OBJ
      FROM ALVW_PARENT_CHILD
      WHERE PARENT_OBJ_TYPE = 'Job'
          AND PARENT_OBJ = 'HR_INITIAL_LOAD'
          AND DESCEN_OBJ_TYPE = 'Dataflow'
  ```

3.8.4 Operational metadata

These tables and views contain information about the run-time statistics of Data Services jobs and data flows.

- *AL_HISTORY* [page 1742]
- *ALVW_FLOW_STAT* [page 1743]

3.8.4.1 AL_HISTORY

This table contains information about the execution statistics of jobs.

Column Name	Description
OBJECT_KEY	Internal ID of the job within the repository.
INST_MACHINE	Computer on which the job was executed.
TYPE	Batch job or a real-time job.
SERVICE	Name of the job.
START_TIME	Time when the job was launched.
END_TIME	Time when the job completed.
EXECUTION_TIME	Difference between start time and end time in seconds.
STATUS	Status of the job upon completion. Examples: Error (E), Success (S).
HAS_ERROR	Displays a zero if there are no errors.

Example

The following query returns the run statistics of all successfully executed jobs:

```
SELECT SERVICE, INST_MACHINE, START_TIME, END_TIME, EXECUTION_TIME
    FROM AL_HISTORY A
    WHERE HAS_ERROR = 0
        AND SERVICE NOT IN ('di_job_al_mach_info', 'CD_JOB_d0cafae2')
```

```
        AND OBJECT_KEY = (SELECT MAX(OBJECT_KEY) FROM AL_HISTORY b WHERE
A.SERVICE = B.SERVICE)
```

3.8.4.2 ALVW_FLOW_STAT

This view contains information about the execution statistics of transforms within data flows.

Column Name	Description
DATAFLOW_NAME	Name of the data flow.
JOB_NAME	Name of the executed job.
JOB_KEY	Identifier that represents a job run. Whenever a job is executed, a new ID is given.
JOB_RUNID	Identifier that represents a single job over its duration. For example, if a job is set to recover from a failed execution and then the job fails, it would restart with the same JOB_RUNID.
RUN_SEQ	Unique identifier for a particular sequence of an execution.
PATH	Position of the transform in the data flow.
OBJECT_NAME	Name of the transform.
OBJECT_TYPE	Type of the transform.
ROW_COUNT	Number of rows processed by this transform.
START_TIME	Time when transform started executing.
END_TIME	Time when transform stopped.
EXECUTION_TIME	The difference between start and end time.

3.9 Data Quality repository statistics tables

3.9.1 Data Quality repository statistics tables

Repository tables hold job processing statistics. There is a table for each aspect of job processing, and the tables contain columns that hold job statistics. There is a set of tables for each of the following Data Quality transforms:

- DSF2 Walk Sequencer
- Geocoder
- Global Address Cleanse
- Match
- USA Regulatory Address Cleanse

Enable statistics in the Data Quality transform settings. Then the software populates applicable repository tables with statistics when you run your job.

Enable reports in the Data Quality transform to obtain specific reports from the job. The software generates reports based on the availability of statistics in the tables. These reports may be regulatory, like the AMAS form for Australia address matching processing, or informative, like the U.S. Addressing Report. You can also create customized reports comprised of the statistics in the repository tables.

Repository tables and related reports

The table below lists the established reports, the corresponding repository table name, and the applicable transforms.

> i Note
>
> There are additional tables that contain data, but they are not related to a specific report.

Report	Repository table	Transform(s)
US CASS report: USPS Form 3553	PSFORM3553DATA	USA Regulatory Address Cleanse
NCOALink Processing Summary report	NCOALINKSUMMARY MEDSTATS NCOACERTIFICATIONDATA	USA Regulatory Address Cleanse
Customer Service Log	CSLSTATS	USA Regulatory Address Cleanse
Broker and List Administrator file	PAFBALASTATS	USA Regulatory Address Cleanse
Delivery Sequence Invoice report	DSFSEQUENCESTATS	DSF2 Walk Sequencer
US Addressing Report	DPVLACSLINKSUMMARY	USA Regulatory Address Cleanse
US Regulatory Locking Report	USREGULATORYLOCKING	USA Regulatory Address Cleanse
Canada SERP report: Statement of Address Accuracy	SERPADDRACCURACY	Global Address Cleanse
Australia AMAS report: Address Matching Processing Summary	AMASADRPROCSUMMARY	Global Address Cleanse
Address Information Code Sample report	ADDRINFOCODEDATA	Global Address Cleanse USA Regulatory Address Cleanse
Address Information Code Summary report	ADDRINFOCODESUMMARY ADDRSTATUSCODEDATA	Global Address Cleanse USA Regulatory Address Cleanse
Address Validation Summary report	ADDRVALIDATESUMMARY	Global Address Cleanse USA Regulatory Address Cleanse
Address Type Summary report	ADDRTYPESUMMARY	Global Address Cleanse USA Regulatory Address Cleanse
Address Standardization Sample	ADDRINFOCODEDATA	Global Address Cleanse

Report	Repository table	Transform(s)
		USA Regulatory Address Cleanse
Geocoder Summary Report (part of the US Addressing report)	GEO_ASSIGN_LEVEL GEO_INFO_CODE	Geocoder transform USA Regulatory Address Cleanse
Address Quality Code Summary	ADDRINFOCODESUMMARY	Global Address Cleanse
Best Record Summary	MTBRINFO MTBRACTION	Match
Match Contribution Report	MTRULESRES MTBRKGRPINFO MTBRKGRP	Match
Match Criteria Summary report	MTCRITINFO MTCRITDEF MTKEYDEF MTCMPCRIT	Match
Match Source Stats Summary report	MTGSSRCSTS MTGSSRCBYSRCSTS	Match
Match Duplicate Sample report	MTDUPESDATA	Match
Match Input Source Output Select Report	MTINSRCINFO	Match
Match Multi-Source Frequency Report	Match statistics tables	Match
New Zealand SOA	SENDRIGHTADDRACCURACY	Global Address Cleanse

Related Information

Match repository statistics tables [page 1772]

3.9.2 Repository tables common columns

There are two columns that are present in most of the repository tables: OBJECT_KEY and OBJECT_ID. These columns contain platform-generated data. These columns are not listed in the descriptions of specific repository tables, but each table contains these columns unless indicated in the individual description.

Column	Data type definition	Description
🔑 OBJECT_KEY	INT	Identification for a specific run (also called Run ID). May not appear in each table.
🔑 OBJECT_ID	NVARCHAR (255)	GUID (globally unique identifier) assigned to a transform. Appears in each table.

3.9.3 Repository tables for USA and Global address cleanse

The table below contains a list of repository tables used for report and statistical information for the USA Regulatory Address Cleanse, Global Address Cleanse, and Geocoder transforms. The sections following this chart contain a topic for each table with descriptions for the fields (columns) in the table.

Repository table name	Description	Transform
ADDRINFOCODEDATA	Contains statistics about each record that generated an address information code during processing. Used for the Address Information Code Sample report.	Global Address Cleanse USA Regulatory Address Cleanse
ADDRINFOCODESUMMARY	Contains statistics about each information code found during processing. Used for the Address Information Code Summary and the Address Quality Code Summary reports.	Global Address Cleanse USA Regulatory Address Cleanse
ADDRSTATUSCODEDATA	Contains status codes found by the transform and their descriptions. Used for the Address Information Code Sample Report.	Global Address Cleanse USA Regulatory Address Cleanse
ADDRTYPESUMMARY	Contains statistics about address types found during processing. Used for the Address Type Summary.	Global Address Cleanse USA Regulatory Address Cleanse
ADDRVALIDATESUMMARY	Contains record validation statistics found by the transform. Used for the Address Validation Summary.	Global Address Cleanse USA Regulatory Address Cleanse
AMASADRPROCSUMMARY	Contains summary statistics for address processing. Used for the AMAS (Australia Matching Approval System) Address Matching Processing Summary.	Global Address Cleanse

Repository table name	Description	Transform
CSLSTATS	Contains statistics from NCOALink processing. Used for the NCOALink CSL (Customer Service Log).	USA Regulatory Address Cleanse
DPVLACSLINKSUMMARY	Contains statistics about the DPV, DSF2, SuiteLink, and LACSLink processing. Used for the U.S. Addressing Report.	USA Regulatory Address Cleanse
DSFAUGMENTSTATS	Contains a detailed record plus the DSF2 licensee name and processing date. Used in the DSF2 Augment Statistics Log File.	USA Regulatory Address Cleanse
DSFSEQUENCESTATS	Contains DSF2 sequence statistics per postcode/sortcode combination. Used in the Delivery Sequence Invoice Report.	USA Regulatory Address Cleanse
MEDSTATS	Contains statistics for NCOALink move effective dates. Used for the NCOALink Report.	USA Regulatory Address Cleanse
NCOALCERTIFICATIONDATA	Contains statistics required for NCOALink processing. Used for the NCOALink Processing Summary Report.	USA Regulatory Address Cleanse
NCOALINKSUMMARY	Contains summary statistics required for NCOALink processing. Used for the NCOALink Processing Summary Report.	USA Regulatory Address Cleanse
PAFBALASTATS	Contains statistics for NCOALink processing. Used for the Processing Acknowledgement Form and the Broker and List Administrator file.	USA Regulatory Address Cleanse
PSFORM3553DATA	Contains data for the USPS Form 3553 report that is submitted with all CASS-certified mailings.	USA Regulatory Address Cleanse
SERPADDRACCURACY	Contains statistics about the accuracy of the addresses in the list. Used for the Statement of Address Accuracy report (Canada).	Global Address Cleanse
SENDRIGHTADDRACCURACY	Contains summary statistics for New Zealand address processing.	Global Address Cleanse

Repository table name	Description	Transform
	Used for the SendRight New Zealand Statement of Accuracy (SOA) report.	
USREGULATORY-LOCKING	Contains information about the record that caused the lock. Used for the US Regulatory Locking Report.	USA Regulatory Address Cleanse

3.9.3.1 ADDRINFOCODEDATA

This table contains statistics about each record that generated an address information code during processing. The information is used for the Address Information Code Sample report. It applies to the Global Address Cleanse and USA Regulatory Address Cleanse transforms.

Column	Data type definition	Description
🔑 RECNUM	INT	Record number that generated the information code.
🔑 NAME	VARCHAR (128)	Name of the field that generated the information code. For example, COUNTRY_CODE, ENGINE_NAME and so on.
VALUE	VARCHAR (256)	Information contained in the field identified under NAME. For example, “US” (COUNTRY_CODE) or “USA” (ENGINE_NAME).

> **Note**
>
> The OBJECT_KEY column is included in this table but it is not listed. it is a primary key common to most tables. The OBJECT_ID column does not apply to this table.

Related Information

Repository tables common columns [page 1745]

3.9.3.2 ADDRINFOCODESUMMARY

This table contains statistics about each information code found during processing. The information is used for the Address Information Code Summary report (USA Regulatory Address Cleanse transform) and the Address Quality Code Summary (Global Address Cleanse transform).

Column	Data type definition	Description
🔑 COUNTRYID	VARCHAR (2)	Country code applicable to the listed INFOCODE.
🔑 INFOCODE	VARCHAR (4)	Information code.
INFOCOUNT	INT	Total number of records in the list that received the listed INFOCODE.
🔑 ENGINENAME	CHAR (50)	Name of the engine applicable to the listed INFOCODE. For the Global Address Cleanse transform, it is the name of the engine that processed the data (USA, CANADA, GLOBAL_ADDRESS, or GLOBAL_ADDRESS_CJK). For the USA Regulatory Address Cleanse transform, it is always USA.
🔑 DATA_SOURCE_ID	VARCHAR (80)	Code that uniquely identifies the list that contains the listed INFOCODE.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.3 ADDRSTATUSCODEDATA

This table contains status codes and descriptions. The information is used for the Status Code Sample Report. It is applicable for Global Address Cleanse and USA Regulatory Address Cleanse transforms.

Column	Data type definition	Description
🔑 RECNUM	INT	Record number that generated the information code.
🔑 NAME	VARCHAR (128)	Name of the field that generated the information code. For example, COUNTRY_CODE, ENGINE_NAME.

Column	Data type definition	Description
VALUE	VARCHAR (256)	Information contained in the field identified under NAME. For example, “US” (COUNTRY_CODE) or “USA” (ENGINE_NAME).

ℹ Note

The OBJECT_KEY column is included in this table but it is not listed. it is a primary key common to most tables. The OBJECT_ID column is not applicable to this table.

Related Information

Repository tables common columns [page 1745]

3.9.3.4 ADDRTYPESUMMARY

This table contains statistics about address types found during processing for the Address Type Summary. The Address Type Summary contains the record count for each Assignment_Type field value (Global Address Cleanse transform) or Address_Type field value (USA Regulatory Address Cleanse transform).

Column	Data type definition	Description
🔑 ENGINENAME	CHAR (50)	Name of the engine used to process the list. For the Global Address Cleanse transform, it is the name of the engine that processed the data (USA, CANADA, GLOBAL_ADDRESS, and GLOBAL_ADDRESS_CJK). For the USA Regulatory Address Cleanse transform, it is always USA.
🔑 COUNTRYID	CHAR (2)	Country code applicable to the listed address type.
BLDNAMERECS	INT	Number of records that were determined to be building name addresses (Global Address Cleanse transform).
FIRMNAMERECS	INT	Number of records that were determined to be firm addresses.
GENDELIVERYRECS	INT	Number of records that were determined to be general delivery addresses.

Column	Data type definition	Description
UNIQSUBURBANRECS	INT	Number of records that were determined to be unique suburban addresses.
MOBILEROUTERECS	INT	Number of records that were determined to be mobile route addresses (Canadian addresses, Global Address Cleanse transform).
HIGHRISERECS	INT	Number of records that were determined to be highrise addresses.
MILITARYRECS	INT	Number of records that were determined to be military addresses.
POSTALRECS	INT	Number of records that were determined to be post office box addresses.
RURALRECS	INT	Number of records that were determined to be rural route addresses.
STREETRECS	INT	Number of records that were determined to be street addresses.
UNDERTERMINDRECS	INT, NULL	Number of records that were determined to be non US records (USA Regulatory Address Cleanse transform).
PROCESSEDRECS	INT	Number of records processed by the transform.
🔑 SUITEL_TYPE	INT	Pre-SuiteLink or post-SuiteLink processing. Values include: 1 (Post-SuiteLink processing) -1 (Pre-SuiteLink processing)
🔑 DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to the list.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.5 ADDRVALIDATESUMMARY

This table contains record validation statistics. The information is used for the Address Validation Summary. It is applicable for Global Address Cleanse and USA Regulatory Address Cleanse transforms.

Column	Data type definition	Description
🔑 ENGINENAME	CHAR (50)	Name of the engine applicable to the COMPONENTNAME column. For the Global Address Cleanse transform, it is the name of the engine that processed the data (USA, CANADA, GLOBAL_ADDRESS, and GLOBAL_ADDRESS_CJK). For the USA Regulatory Address Cleanse transform, it is always USA.
🔑 COUNTRYID	CHAR (2)	Country code applicable to the COMPONENTNAME column.
🔑 COMPONENTNAME	NVARCHAR (64)	Address component (ADDRESS1, LOCALITY1, and so on) to which the counts and percentages apply.
ADDEDRECS	INT	Number of records processed by the transform.
DELETEDRECS	INT	Number of records that were deleted during processing for the applicable COMPONENTNAME.
CORRECTEDRECS	INT	Number of records that were corrected during processing for the applicable COMPONENTNAME.
UNCHANGEDRECS	INT	Number of records that remained unchanged for the applicable COMPONENTNAME.
🔑 SUITEL_TYPE	INT	Indicates whether the count information applies to pre-SuiteLink or post-SuiteLink processing, or pre-NCOALink or post-NCOALink processing. Values include: 1 (post-SuiteLink/NCOALink processing) -1 (pre-SuiteLink/NCOALink processing)
🔑 DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to the list.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.6 CSLSTATS

This table contains statistics for NCOALink processing for the Customer Service Log. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
🔑 LICENSEEID	NVARCHAR (4)	NCOALink licensee's identification number assigned by the USPS.
🔑 SEQNUM	INT	DSF2 sequence number assigned to the address.
DTLRECORD	NVARCHAR (1000)	String of Customer Service Log field values.
LICENSEETYPE	NVARCHAR (1)	Type of service provider. Values are: E (End user) F (Full service provider) L (Limited service provider)
PROCESSINGDATE	DATETIME	Date and time the job was processed through the software in YYYY-MM-DD HR:MIN:SEC format.
RESERVED_COUNT1	INT	Reserved for future record counts.
RESERVED_COUNT2	INT	Reserved for future record counts.
RESERVED_COUNT3	INT	Reserved for future record counts.
RESERVED_STRING1	NVARCHAR (100)	Reserved for future strings.
RESERVED_STRING2	NVARCHAR (100)	Reserved for future strings.
RESERVED_STRING3	NVARCHAR (100)	Reserved for future strings.
🔑 DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to the list.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.7 DPVLACSLINKSUMMARY

This table contains statistics about the DPV, DSF2, LACSLink, and SuiteLink processing. The information is used in the US Addressing Report. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
LACSCONVERTED_A	INT	Number of LACSLink records that were converted by the transform (return code A).
LACSCONVERTED_92	INT	Number of LACSLink records that matched after dropping the secondary number from the input address (return code 92).
LACSNOTCONVERTED_09	INT	Number of LACSLink records that matched an input address to an old address, and the old address is a high-rise default address. No new addresses provided (return code 09).
LACSNOTCONVERTED_00	INT	Number of records that had no match to LACSLink and, therefore, no addresses converted (return code 00).
LACSNOTCONVERTED_14	INT	Number of LACSLink records that matched to LACSLink, but couldn't be converted to a deliverable address (return code 14).
DPVVALIDATED_Y	INT	Number of records that were validated for DPV (status of Y, primary and secondary range is valid).
DPVVALIDATED_S	INT	Number of records that contained an invalid secondary range (DPV status of S, secondary range not present).
DPVVALIDATED_D	INT	Number of records that did not have a secondary range (DPV status of D, secondary range not present).
DPVVALIDATED_CMRA	INT	Number of records that were validated as a CMRA (Commercial Mail Receiving Agency).
DPVNOTVALIDATED	INT	Number of records that were not validated for DPV.

Column	Data type definition	Description
🔑 SUITEL_TYPE	INT	Indicates whether the information is from pre-SuiteLink or post-SuiteLink processing. 1 (Pre-SuiteLink processing) -1 (Post-SuiteLink processing) 0 (NCOALink is not enabled so there is no pre-NCOALink or post-NCOALink processing sections)
SUITEL_MATCH_A	INT	Number of records that matched to SuiteLink and had secondary information added.
SUITEL_NOMATCH_00	INT	Number of records that did not match to SuiteLink.
DPVVALIDATED_VACANT	INT	Number of records with DPV status of vacant.
DPVVALIDATE_NOSTATS	INT	Number of records with DPV status of nostats.
DSF2_DROP	INT	Number of records that are dropped at a delivery point that serves businesses or families (for example, a CMRA).
DSF2_BUSINESS	INT	Number of records that have a business address.
DSF2_THROWBACK	INT	Number of records that are throwbacks (customer wants delivery at PO Box instead of street address).
DSF2_SEASONAL	INT	Number of records that are seasonally occupied.
DSF2_EDUCATIONAL	INT	Number of records that are educational institutions.
DSF2_CURB	INT	Number of records that have a curb-side delivery indicator.
DSF2_CENTRAL	INT	Number of records that have a central delivery indicator.
DSF2_DOOR	INT	Number of records that have a door-slot delivery indicator.
DSF2_NDCBU	INT	Number of records that have an NDCBU (Neighborhood Delivery Centralized Box Unit) delivery indicator.
🔑 DATA_SOURCE_ID	NVARCHAR (80)	A unique identification for a list.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.8 DSFAUGMENTSTATS

This table contains information used in the DSF2 Augment Statistics Log file. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
🔑 LICENSEE_NUMBER	NVARCHAR (4)	DSF2 licensee number.
🔑 SEQNUM	INT	Sequence number from 0000 to 9999. ● If the Postcode2 field is blank, this column is blank. ● If the Postcode2 field is not blank and the DPV_Status is not Y, then the column contains “0000”.
LICENSEE_NAME	NVARCHAR (40)	DSF2 licensee name.
PROCESSING_DATE	DATETIME	Date and time the job was processed through the software in YYYY-MM-DD HR:MIN:SEC format.
DTLRECORD	NVARCHAR (1000)	String of log file field values for DSF2 Augment Statistics.
🔑 DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to the list.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.9 DSFSEQUENCESTATS

This table contains information for the Delivery Sequence Invoice Report. It is applicable for the USA Regulatory Address Cleanse transforms.

Column	Data type definition	Description
🔑 POSTCODE1	NVARCHAR (5)	Postcode (ZIP Code) for the records being processed.
🔑 SORTCODE_ROUTE	NVARCHAR (4)	Sortcode route (carrier route) for the records being processed.
🔑 LICENSEE_NAME	NVARCHAR (40)	DSF2 licensee name.
TOTAL_DELIVERY_POINTS	INT	Number of deliveries within the specified postcode/sortcode combination.
TOTAL_RESIDENCES	INT	Number of residences within the specified postcode/sortcode combination.
TOTAL_DELIVERY_POINTS_SEQ	INT	Number of delivery points sequenced by the transform for the specific postcode/ sortcode combination.
TOTAL_RESIDENCES_SEQ	INT	Number of residences sequenced for the specific postcode/sortcode combination.
SITE_LOCATION	NVARCHAR (20)	Location of the site where the DSF2 walk sequence processing occurred.
LIST_ID	NVARCHAR (6)	Unique ID assigned by the licensee to identify the list.
PROCESSING_DATE	DATETIME	Date and time the job was processed through the software in YYYY-MM-DD HR:MIN:SEC format.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.10 MEDSTATS

This table contains statistics for NCOALink move effective dates. The information is used for the NCOALink Report. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description

🔑 CATEGORY	NVARCHAR (1)	Return code category:
		A = Return codes A, 91, 92
		B = Return codes 01, 02, 03
		C - Return codes 05, 14, 19

🔑 PERIOD	NVARCHAR (2)	Time period for the number of records for each return code category. Values include the following: 1 = Months 0-3 2 = Months 4-6 3 = Months 7-12 4 = Months 13-18 5 = Months 19+
MEDCOUNT	INT	Number of moves in the specified period.
🔑 DATA_SOURCE_ID	NVARCHAR (80)	Unique identification for a list.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.11 NCOALCERTIFICATIONDATA

This table contains statistics from NCOALink processing and is used for the NCOALink Reports. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
MAILER_PAF_ID	VARCHAR (40)	Combination of the following information: • USPS-assigned license ID (first 4 characters) • List Owner NAICS Code (next 5 characters) • Frequency of processing (next 2 characters) • Licensee-assigned List ID (last 6 characters)
MAILER_COMPANY	VARCHAR (30)	Name of the customer that requested NCOALink processing.
PROCESSES_USED	VARCHAR (768)	All USPS processes used to obtain the final data results.
DATA_RETURNED	VARCHAR (1)	Purpose for NCOALink processing. C (Change of address) F (Return codes) S (Statistics)
PRE_PROCESSES	VARCHAR (1)	Indicates that the list was pre processed and what data modifications ocurred. Values are: N (None) Y (Yes, but with no data modifications). D (Yes, data modifications from sources other than postal data). P (Yes, data modifications from postal data only. For example, ZIP+4 or DPV) B (Yes, data modifications from postal and other sources)
POST_PROCESSES	VARCHAR (1)	Indicates that the list was post processed and what data modifications ocurred. Values are: N (None). Y (Yes, but with no data modifications). D (Yes, data modifications from sources other than postal data). P (Yes, data modifications from postal data only. For example, LACSLink).

<table>
<tr><th>Column</th><th>Data type definition</th><th>Description</th></tr>
<tr><td></td><td></td><td>B (Yes, data modifications from postal and other sources).</td></tr>
<tr><td>CONCURRENT_PROCESSES</td><td>VARCHAR (1)</td><td>Indicates that the list was concurrently processed and the data modifications that ocurred. Values are:
N (None).
Y (Yes, but with no data modifications).
D (Yes, data modifications from sources other than postal data).
P (Yes, data modifications from postal data only. For example, ZIP+4 or DPV).
B (Yes, data modifications from postal and other sources).</td></tr>
<tr><td>LIST_NAME</td><td>VARCHAR (30)</td><td>Name of the list being processed.</td></tr>
<tr><td>LICENSEE_ID</td><td>VARCHAR (4)</td><td>Unique ID for the broker or list administrator.</td></tr>
<tr><td>LICENSEE_COMPANY</td><td>VARCHAR (50)</td><td>Broker or list administrator name.</td></tr>
<tr><td>PROCESSING_CATEGORY</td><td>VARCHAR (20)</td><td>Reason for the NCOALink processing. Values are:
<table>
<tr><th>Option</th><th>Printed on report</th></tr>
<tr><td>Marketing</td><td>MKTG TEST</td></tr>
<tr><td>Normal</td><td>NORMAL</td></tr>
<tr><td>Stage I</td><td>STAGE I</td></tr>
<tr><td>Stage II</td><td>STAGE II</td></tr>
<tr><td>System testing</td><td>SYS TEST</td></tr>
</table></td></tr>
<tr><td>SERVICE_PROVIDER</td><td>VARCHAR (1)</td><td>NCOALink list processor's provider level. Values are:
0 End user
1 Limited service provider
2 Full service provider</td></tr>
<tr><td>NCOAL_PROCESS_DATE</td><td>VARCHAR (10)</td><td>Date based on the time that the list was processed. Automatically generated by the software, and included in reports.</td></tr>
<tr><td>LIST_RETURN_DATE</td><td>VARCHAR (10)</td><td>Date obtained from the <i>List returned date</i> option in the USPS License Information group.</td></tr>
</table>

Column	Data type definition	Description
DPV_ENABLED	VARCHAR (1)	DPV is enabled in the transform (Y/N indicator in reports).
LACSL_ENABLED	VARCHAR (1)	LACSLink is enabled in the transform (Y/N indicator in reports).
SUITEL_ENABLED	VARCHAR (1)	SuiteLink is enabled in the transform (Y/N indicator in reports).
ANK_ENABLED	VARCHAR (1)	ANKLink is enabled in the transform (Y/N indicator in reports).
USE_BUSINESS_MOVES	VARCHAR (1)	Business moves were included in processing (Y/N indicator in reports).
USE_INDIVIDUAL_MOVES	VARCHAR (1)	Individual moves were included in processing (Y/N indicator in reports).
USE_FAMILY_MOVES	VARCHAR (1)	Family moves were included in processing (Y/N indicator in reports).
MAIL_CLASS	VARCHAR (1)	Mail class that was processed.
MATCH_LOGIC	VARCHAR (1)	Move types that were processed. Values are: S (Standard move types such as business, individual, and family matches). I (Individual only). B (Business only). C (Individual and business only).
STD_OUTPUT_RETURNED	VARCHAR (1)	Transform-generated value that indicates whether the standard output was returned. Values are: Y (All NCOALink-required output returned to client). N (Post processes modified return information. For example updates applied to list). B (Post processes modified return information. However, a separate file containing all required output data was also returned).
PROCESS_PERIODICAL	VARCHAR (1)	Periodicals mail was processed by the transform. (Y/N indicator in reports.)
PROCESS_FIRST	VARCHAR (1)	First Class mail was processed by the transform. (Y/N indicator in reports.)

Column	Data type definition	Description
PROCESS_STD	VARCHAR (1)	Standard mail was processed by the transform. (Y/N indicator in reports.)
PROCESS_PACKAGE	VARCHAR (1)	Package Services mail was processed by the transform. (Y/N indicator in reports.)
TOTAL_RECORDS	INT	Number of records processed by the transform.
NCOAL_MATCHES	INT	Number of NCOALink matches found in the list.
ANK_MATCHES	INT	Number of ANKLink matches found in the list.
ZIP4_MATCHES	INT	Number of ZIP+4 matches found in the list.
DPV_MATCHES	INT	Number of DPV matches found in the list.
LACSL_MATCHES	INT	Number of LACSLink matches found in the list.
SUITEL_MATCHES	INT	Number of SuiteLink matches found in the list.
INDIVIDUAL_RECORDS	INT	Number of individual type records found in the list.
FAMILY_RECORDS	INT	Number of family type records found in the list.
BUSINESS_RECORDS	INT	Number of business type records found in the list.
NCOAL_NAME	VARCHAR (16)	Name of the NCOALink-certified software.
NCOAL_VERSION	VARCHAR (16)	Version of the NCOALink-certified software.
RESERVED_COUNT1	INT	Extra field for additional counts.
RESERVED_COUNT2	INT	Extra field for additional counts.
RESERVED_COUNT3	INT	Extra field for additional counts.
RESERVED_STRING1	VARCHAR (100)	Extra field for additional string.
RESERVED_STRING2	VARCHAR (100)	Extra field for additional string.
RESERVED_STRING3	VARCHAR (100)	Extra field for additional string.
🔑 DATA_SOURCE_ID	VARCHAR (80)	Unique identification code assigned to the list.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.12 NCOALINKSUMMARY

This table contains summary statistics required for NCOALink processing. The information is used for the NCOALink Summary Report. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
🔑 CODE	NVARCHAR (5)	Return code.
CODEGROUP	NVARCHAR (1)	Group in which the code applies. 1 = New address provided by NCOALink 2 = Found COA 3 = Cannot match COA 4 = From the daily delete process
CODEDESC	NVARCHAR (50)	Description for the return code listed.
RECCOUNT	INT	Number of list records found for the code listed.
ADDRESS_PROVIDED	NVARCHAR (1)	Indicates whether an address was provided (Y/N).

ADDRESS_SOURCE	NVARCHAR (1)	D = Derived by data
		S = Derived from software

DETAILED_DESC	VARCHAR (1)	Provides a more detailed description of the listed code.
🔑 DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to a list.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.13 PAFBALASTATS

This table contains statistics for NCOALink processing. The information is used for the Processing Acknowledgement Form and the Broker and List Administrator file. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
🔑 LICENSEEID	NVARCHAR (4)	Service provider's NCOALink license ID.
DTLRECORD	NVARCHAR (700)	All field values per list.

🔑 LOGTYPE	NVARCHAR (1)	The type of log file:
		P = PAF
		B = BALA

🔑 SEQNUM	INT	DSF2 sequence number assigned to the address.
RESERVED_COUNT1	INT	Reserved for future record counts.
RESERVED_COUNT2	INT	Reserved for future record counts.
RESERVED_COUNT3	INT	Reserved for future record counts.
RESERVED_STRING1	NVARCHAR (255)	Reserved for future text.
RESERVED_STRING2	NVARCHAR (255)	Reserved for future text.
RESERVED_STRING3	NVARCHAR (255)	Reserved for future text.
🔑 DATA_SOURCE_ID	NVARCHAR (80)	Unique identification for a list.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.14 PSFORM3553DATA

This table contains data for the USPS Form 3553 report that is submitted with all CASS-certified mailings.

Column	Data type definition	Description
CASSCOMPANYNAME	NVARCHAR (50)	Name of the company that is CASS-certified.

Column	Data type definition	Description
CASSCONFIGURATION	NVARCHAR (10)	Software configuration settings as they appear on the CASS certificate.
CASSSOFTWARE	NVARCHAR (60)	Name and version of the software that is CASS certified.
Z4COMPANYNAME	NVARCHAR (40)	Name of the company that is Z4Change certified.
Z4CONFIGURATION	NVARCHAR (10)	Software configuration settings for Z4Change.
Z4SOFTWARE	NVARCHAR (60)	Name and version of the software that is certified for Z4Change processing.
eLOTCOMPANYNAME	NVARCHAR (40)	Name of the company that is certified for eLOT processing.
eLOTCONFIGURATION	NVARCHAR (10)	Software configuration settings for eLOT.
eLOTSOFTWARE	NVARCHAR (60)	Name and version of the software that is certified for eLOT processing.
MASSCOMPANYNAME	NVARCHAR (40)	Not populated from the software.
MASSCONFIGURATION	NVARCHAR (10)	Not populated from the software.
MASSSOFTWARE	NVARCHAR (60)	Not populated from the software.
MLOCRSERIALNUMBER	NVARCHAR (20)	Not populated from the software.
LISTPROCESSOR	NVARCHAR (35)	MASS list processor. Not populated from the software.
ZIP4DBDATE	VARCHAR (15)	ZIP4 directory date.
Z4CHANGEDATE	VARCHAR (15)	Not populated from the software.
Z4CHANGEDBDATE	VARCHAR (15)	Z4Change directory date.
eLOTDATE	VARCHAR (15)	Not populated from the software.
eLOTDBDATE	VARCHAR (15)	eLOT directory date.
CRISDATE	VARCHAR (15)	Not populated from the software.
CRISDBDATE	VARCHAR (15)	Not populated from the software.
ADDRESSLISTNAME	NVARCHAR (20)	Name of the address list processed. (Entered in *CASS Report Options* > *List Name*.)
NUMBEROFLISTS	INT	Not populated from the software.
TOTALRECPROCESSED	INT	Number of records processed in the job.
ZIP4RECORDSCODED	INT	Number of ZIP+4 records that were DPV confirmed.
Z4CHGRECORDSCODED	INT	Number of records that were Z4Change coded.

Column	Data type definition	Description
DPBRECORDSCODED	INT	Number of records with delivery point bar codes.
DPBCDATEFROM	VARCHAR (15)	Not populated from the software.
FIVEDIGITRECCODED	INT	Number of records that were assigned 5-digit ZIP Codes.
FIVEDIGITDATEFROM	VARCHAR (15)	Not populated from the software.
CRRTRECORDSCODED	INT	Number of records that were assigned carrier route codes.
CRRDATEFROM	VARCHAR (15)	Not populated from the software.
eLOTRECORDSCODED	INT	Number of records that were assigned eLOT codes.
eLOTDATEFROM	VARCHAR (15)	Not populated from the software.
MAILERINFO1	NVARCHAR (50)	Information entered in the USA Regulatory Address Cleanse transform in the CASS report options group.
MAILERINFO2	NVARCHAR (50)	Information entered in the USA Regulatory Address Cleanse transform in the CASS report options group.
MAILERINFO3	NVARCHAR (50)	Information entered in the USA Regulatory Address Cleanse transform in the CASS report options group.
MAILERINFO4	NVARCHAR (50)	Information entered in the USA Regulatory Address Cleanse transform in the CASS report options group.
HIGHRISEDEFAULT	INT	Number of records that were assigned as highrise defaults.
HIGHRISEEXACT	INT	Number of records that were assigned as highrise exact matches.
RURALROUTEDEFAULT	INT	Number of records that were assigned as rural route default matches.
RURALROUTEEXACT	INT	Number of records that were assigned as rural route exact matches.
LACS	INT	Number of addresses that were converted through the LACSLink process.
EWS	INT	Number of records that were assigned as EWS addresses (and, therefore, are not listed in the current U.S. Postal Service® ZIP + 4 File).
DPV	INT	Number of records that were confirmed as ZIP + 4/DPV that matched to a highrise default, and the SuiteLink process returned the appropriate suite number.

Column	Data type definition	Description
RDI	INT	Not populated from the software (always zero).
RESERVED_COUNT1	INT	Reserved for future record counts.
RESERVED_COUNT2	INT	Reserved for future record counts.
RESERVED_COUNT3	INT	Reserved for future record counts.
RESERVED_STRING1	NVARCHAR (100)	Reserved for future strings.
RESERVED_STRING2	NVARCHAR (100)	Reserved for future strings.
RESERVED_STRING3	NVARCHAR (100)	Reserved for future strings.
🔑 DATA_SOURCE_ID	NVARCHAR (80)	Unique identification assigned to the list.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.15 SENDRIGHTADDRACCURACY

This table contains summary statistics about the address processing for the Global Address Cleanse SendRight certification report. The information is used for the SendRight Address Accuracy report (for New Zealand). These statistics are generated by the Global Address Cleanse transform.

Column	Data type definition	Description
CERTCOMPANYNAME	VARCHAR 40	Name of the company that is SendRight certified. This value will be SAP Business Objects.
CERTPRODNAME	VARCHAR 40	Name of the product that is SendRight certified. This value will be Data Services.
CERTPRODVER	VARCHAR 20	Version of the SendRight-certified software.
PAFVERSION	VARCHAR 20	Version of the current (most recent) PAF.

Column	Data type definition	Description
		PAF refers to the version number supplied by New Zealand post.
SOAUNIQUEID	VARCHAR 20	Unique ID generated by the certified engine to guarantee a unique report for each Global Address Cleanse transform within a data flow.
CUSTOMERNAME	VARCHAR 40	Name of the list owner or list processor.
MAILERADDR1–6	VARCHAR 60	Name and address of the person or organization for whom you are preparing the mailing. (6 columns, 60 varchars each).
FILENAME	VARCHAR 40	Input file name.
TOTALRECPROC	INT	Number of records processed.
UNIQUEMATCHES	INT	Number of unambiguous matches between the input addresses and one address record in the PAF.
BASEMATCHES	INT	Number of unambiguous matches between the input addresses and one base address.
DATEISSUED	DATETIME	Date that this report was generated.
DATEEXPIRED	DATETIME	Date that this report will expire. (Always exactly 1 year after generation date.)
TOTALVALIDATED	INT	Number of records validated.
ADDRESSACCURACY	VARCHAR 8	Percentage of the total records validated from the total number of records processed.
RESERVED_COUNT1–3	INT	Reserved for future use.
RESERVED_STRING1–3	VARCHAR 255	Reserved for future use.

3.9.3.16 SERPADDRACCURACY

This table contains statistics about the accuracy of the addresses in the list. The information is used for the SERP (Statement of Address Accuracy) report (Canada). These statistics are generated by the Global Address Cleanse transform.

Column	Data type definition	Description
VENDORNAME	VARCHAR (40)	Name of the Address Accuracy vendor. This value will be SAP Business Objects.
VENDORADDR1-2	VARCHAR (40)	Address of the Address Accuracy vendor. This will be the current SAP Business Objects address. (Two columns, 40 varchars each.)
PRODNAMEVER	VARCHAR (36)	Name of the Address Accuracy software product and version. This is hardcoded to the current product and version.
SREXPIRYDATE	VARCHAR (11)	Expiration date of the software's SERP certification.
CUSTOMERNAME	VARCHAR (40)	Company name of the organization for whom the mailing is being prepared.
CUSTOMERADDR1-4	VARCHAR (40)	Name and address of the person or organization for whom the mailing is being prepared. (Four columns, 40 varchars each.)
CUSTOMERCPCNUM	VARCHAR (15)	Date of the postalcode file.
CPCMASTERFILEVER	VARCHAR (11)	Customer's CPC number that is located in the Canada Post Contract.
TOTALRECPROC	INT	Number of urban and rural records processed.
RECASSIGNED	INT	Number of records assigned.
QUEST_RECS_RURAL	INT	Number of questionable rural records.
QUEST_RECS_APPT	INT	Number of questionable apartment records.
RESERVED_COUNT1–3	INT	Reserved for future record counts.
RESERVED_STRING1–3	NVARCHAR (255)	Reserved for future strings.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.3.17 USREGULATORYLOCKING

This table contains information about the record that caused DPV or LACSLink locking. The information is used for the USA Regulatory Locking Report.

Column	Data type definition	Description
🔑 LOCKMODE	INT	Whether the lock was for DPV (1) or LACSLink (2).
LOCK_CODE	VARCHAR (81)	Lock code issued by the software and found in the Alert area of the report.
POSTCODE	VARCHAR (81)	Input postalcode from the address that caused the lock.
PRIMARY_NUMBER	VARCHAR (81)	Input primary range from the address that caused the lock.
PRIMARY_NAME	VARCHAR (81)	Primary name from the address that caused the lock.
UNIT_NUMBER	VARCHAR (81)	Secondary range for the address that caused the lock.
UNIT_DESCRIPTION	VARCHAR (81)	Unit designator for the address that caused the lock.
PRIMARY_PREFIX	VARCHAR (81)	Predirectional for the address that caused the lock.
ADDRESS_TYPE	VARCHAR (81)	Suffix for the address that caused the lock.
PRIMARY_POSTFIX	VARCHAR (81)	Postdirectional for the address that caused the lock.
RESERVED_COUNT1	INT	Reserved for future record counts.
RESERVED_COUNT2	INT	Reserved for future record counts.
RESERVED_COUNT3	INT	Reserved for future record counts.
RESERVED_STRING1	NVARCHAR (255)	Reserved for future strings.
RESERVED_STRING2	NVARCHAR (255)	Reserved for future strings.
RESERVED_STRING3	NVARCHAR (255)	Reserved for future strings.
🔑 DATA_SOURCE_ID	NVARCHAR (80)	Unique identifier for the list.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.4 Geocoder repository statistics tables

The table below contains a list of repository tables used for report and statistical information for the Geocoder transform. The sections following this chart contain a topic for each table with descriptions for the fields (columns) in the table.

Repository table name	Description	Transform
GEO_ASSIGN_LEVEL	Contains Geo statistics per assignment level. Used in the Geocoder Summary Report, a subsection of the US Addressing Report.	Geocoder
GEO_INFO_CODE	Contains Geocoder information codes and record counts per code. Used in the US Addressing Report.	Geocoder

3.9.4.1 GEO_ASSIGN_LEVEL

Statistics generated in this repository table are found in the Geocoder Summary Report. The Geocoder transform is used in conjunction with the Global Address Cleanse transform or the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
🔑 COUNTRY	CHAR (2)	Three-character ISO country code.
🔑 CODE	NVARCHAR (4)	Code that provides information about the geocoding results. Values are: PRE (Primary number exact) PRI (Primary number interpolated) PF (Postcode full) P2P (Postcode2 partial) P1 (Postcode1) L4 (Locality4) L3 (Locality3) L2 (Locality2) L1 (Locality1)
ASSIGN_COUNT	INT	Number of records for the assignment level listed.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Reference Guide: Geocoder fields, Output fields [page 1186]
Repository tables common columns [page 1745]

3.9.4.2 GEO_INFO_CODE

The table below contains statistics about information code found during processing for the Geocoder transform. The Geocoder transform that is used in conjunction with the Global Address Cleanse transform or the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
🔑 INFOCODE	NVARCHAR (4)	A three-character code that provides information about the geocoding results. The status for address and point-of-interest geocoding assignment is indicated in the third character. The status for reverse geocoding assignment is indicated in the second and third characters. If assigned to the best level, the Info_Code field is blank. The first character is not used at this time.
INFO_COUNT	INT	Number of records for the listed INFOCODE.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Output fields [page 1186]
Repository tables common columns [page 1745]

3.9.5 Match repository statistics tables

There are many Match statistics that appear in various reports. These statistics are stored in statistics tables in the repository. The Match repository statistics tables are listed alphabetically below with a brief description for each table. Later, each table is described including the field names, data-type definitions, and descriptions.

Table	Description
MTBRACTION	Information about the best record action. Related report: Best Record Summary report.
MTBRINFO	Best record information. Related report: Best Record Summary report.
MTBRKGRP	Break group information. Related report: Break Group Contribution section of the Match Contribution report.
MTBRKGRPINFO	Information about the top and bottom ten break group strings and counts. Related report: Match Contribution report.
MTCMPCRIT	Match level option setting information for each match set. Related report: Match Level Options section of the Match Criteria Summary report.
MTCRITDEF	Match criteria information for each match set. Related report: Match Input Fields and Detailed Criteria Definition sections of the Match Criteria Summary report.
MTDUPESDATA	Information about match records as a sample of the match results. Related report: Duplicate Sample report.
MTGSSRCBYSRCSTS	Inter and intra-source match counts. Related report: Match Source Statistics Summary report.
MTGSSRCSTS	Information about the distribution of the matches found in various input source records including how the matches were distributed as master records and subordinate records. Related report: Match Source Statistics report.
MTINFO	Name of the match set. Related reports: All of the Match reports except the Best Record Summary report.
MTINSRCBYSRC	Information about how often each input source matched the other input sources.
MTINSRCGRPINFO	Input source group information. The table is populated once per transform if statistics are enabled and the source groups section of the input source object is defined.
MTINSRCINFO	Input source information. The table is populated once per transform if statistics are enabled and the input source object is defined.
MTINSRCMSRC	Multi-source statistics of each input source and each source group. The table is populated once per input source group statistics object if statistics are enabled.

Table	Description
MTINSRCSELECT	Input source group information. The table is populated once per input source select record object if statistics are enabled.
MTINSRCSTATS	Statistics of each input source and each source group. The table is populated once per input source group statistics object if statistics are enabled.
MTKEYDEF	Preprocessing criteria information for each key field. Related report: Match Input Fields section of the Match Criteria Summary report.
MTPROCESS	Information about match processing. Related reports: All Match reports except the Best Record Summary report.
MTRULESRES	Information about the effect of the criteria on the total matching process. Related report: Match Contribution report (Criteria Information sub report).

3.9.5.1 MTBRACTION

This table contains best record information and is applicable to the Match transform. The information is used for the Best Record Summary report.

> Note
>
> This table is populated only if the Best Record functionality is enabled in your job.

Column	Data type definition	Description
🔑 PROCID	INT	Sequential number that identifies a match level or an association.
🔑 ACTIONID	INT	Sequential number that identifies a Best Record Action section.
BRNAME	NVARCHAR (15)	Name of the Best Record operation you specified in the Match Editor.
SRCFLD	NVCHAR (256)	Source field used in the Best Record Action section. Note This column is blank if a source expression is completed instead of a source field.
DSTFLD	NVCHAR (256)	Destination field used in the Best Record Action section.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.2 MTBRINFO

This table contains best record information and is applicable to the Match transform. The information is used for the Best Record Summary report.

ℹ Note

This table is populated only if the Best Record functionality is enabled in your job.

Column	Data Type Definition	Description
🔑 PROCID	INT	Sequential number that identifies a match level or an association.
🔑 BRNAME	NVARCHAR (255)	Name of the best record object.
POSTDEST	CHAR (1)	Destination for the post. Values are A (all), M (master only), or S (subs only).
POSTOPD	CHAR (1)	More than one posting per destination will be attempted or not for each record (Y/N).
PROTECTDROPS	INT	Number of post operations that were canceled because the destination record was protected.
DSTFLDDROPS	INT	Number of post operations that were canceled because the destination record had already been posted to once, and POSTOPD was set to Y.
FILTERDROPS	INT	Number of post operations that were canceled because the Best Record filter returned F (false).
POSTCOMPLETS	INT	Number of post operations that were successfully completed.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.3 MTBRKGRP

This table contains information about match break groups and is applicable to the Match transform. The information is used in the Match Contribution Report. This table is generated if matching is performed (not associating).

ℹ Note

If breaking is not defined in the job setup, then all records are included in one break group.

Column	Data type definition	Description
NUMRECS	INT	Number of records processed in all of the break groups.
ELAPSEDTIME	INT	Elapsed time to process the break groups (in seconds).
CMPBUFMAXRECS	INT	Maximum number of records that can fit into the compare buffer at one time.
NOBRKCOMPARES	FLOAT	Number of comparisons that would be made without using any break group strategy (or putting all records in a single break group).
BRKGRPCOUNT	INT	Number of break groups formed based on the break group strategy.
BRKGRPLARGEST	INT	Largest break group processed.
BRKGRPCOMPARES	FLOAT	Number of comparisons made in all the break groups.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.4 MTBRKGRPINFO

This table contains information about the top and bottom ten break group strings and counts and is applicable to the Match transform. The information is used in the Break Group Contribution section of the Match Contribution Report.

> **Note**
>
> This table is populated only if the software performs matching and breaking.

Column	Data type definition	Description
🔑 BRKID	INT	Identification number for the break group.
BRKSTR	NVARCHAR (256)	Break string from the break group.
NUMRECS	INT	Number of records in the break group.

> **Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.5 MTCMPCRIT

This table contains match level option setting information for each match set and is applicable to the Match transform. This information is in the Match Level Options section of the Match Criteria Summary report.

Column	Data type definition	Description
🔑 CMPCRITID	INT	Sequential number that identifies the criteria of a match level.
NAME	NVARCHAR (256)	Match level name.

Column	Data type definition	Description
WTMTSCORE	INT	Minimum weighted match score needed to make a match decision.
NUMNAMEMUSTMT	CHAR (1)	Number of names that must match. Values are O (one name) or A (all names).
CMPFTOMNAME	CHAR (1)	Specifies the setting for the *Compare Given Name1 to Given Name2* option. (Y/N)
MTONHYPLNAME	CHAR (1)	Specifies the setting for the *Match on hyphenated family name* option. (Y/N)
TRNONMAIDENADJ	CHAR (1)	Specifies the setting for the *Ignore family name when female* option. (Y/N)
IGNFIRMIFNAME	CHAR (1)	Specifies the setting for the *Ignore Firm if Name matches* option. (Y/N)
IGNORESTIFBOX	CHAR (1)	Specifies the setting for the *Match on Street and RR, or on Box* option. (Y/N)
ADDRBLMIFFIRM	CHAR (1)	Specifies the setting for the *Address matches blank if Firms match* option. (Y/N)
UNIQRESRRNOBOX	CHAR (1)	Specifies the setting for the *Unique on resident if RR, but no Box* option. (Y/N)

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.6 MTCRITDEF

This table contains match criteria information for each match set and is applicable to the Match transform. The information is used in the Match Input Fields and Detailed Criteria Definition sections of the Match Criteria Summary.

Column	Data type definition	Description
🔑 CMPCRITID	INT	Sequential number that identifies the criteria of a match level.
🔑 CRITID	INT	Sequential number that identifies an individual criteria.
CRITNAME	NVARCHAR (256)	Name given to the match criteria when it was created.
KEYID	INT	Sequential number that identifies the type of a key field.
MTSCORE	INT	Threshold for similarity scores. Similarity scores at or above this setting are considered a match.
NOMTSCORE	INT	Threshold for similarity scores. Similarity scores at or below this setting are not considered a match.
ONEFLDBLNKOP	CHAR (1)	Setting for the One Field Blank operation. Values are E (evaluate) and I (ignore).
ONEFLDBLNKSCORE	INT	Score given to the criteria when one of the two fields compared is blank.
BTFLDBLNKOP	CHAR (1)	Setting for the Both Fields Blank operation. Values are E (evaluate) and I (ignore).
BTFLDBLNKSCORE	INT	Score given to the criteria when both fields compared are blank.
CNTRBTOWTSCORE	INT	Weight percentage given to the match criteria's contribution to the weighted score.
USEINWTSCOREIFGT	INT	Minimum similarity score needed to qualify the match criteria for use in determining the weighted match score.
ZWTSCOREIFLTOREQ	INT	Minimum similarity score needed for the match criteria to qualify for contributing a value other than zero to the weighted match score.
CMPALGO	CHAR (1)	String comparison algorithm that was used. Values are F for field or W for word.
CHKTRANSPSLET	CHAR (1)	Indicates whether to check for transposed letters (Y/N).
INITADJSCORE	INT	Adjustment score given when fields with whole words match to fields with initials.
SUBSTRADJSCORE	INT	Adjustment score given when fields with longer strings of words match to fields with shorter strings of words (the

Column	Data type definition	Description
		shorter string must match the first part of the longer string).
APPRSUBADJSCORE	INT	Adjustment score given when fields with longer strings of words do not match to fields with shorter strings of words (the shorter string does not match the first part of the longer string).
ABBRADJSCORE	INT	Adjustment score given to the abbreviation substring adjustment score when the first letter of the shorter word matches the first letter of the longer word, and all remaining letters of the shorter word appear in the longer word in the same order as in the shorter word.
EXTABBRADJSCORE	INT	Adjustment score given when two fields match based on a combination of the abbreviation adjustment score. Keep in mind the following requirements for the extended abbreviation adjustment: ● The first letter of the short word must match the first letter of the first word in the multiple-word string. The remaining letters of the short word must be found in order in the multiple-word string. ● Letters that match are given a score of 100. The remaining letters are given the score that you specify. ● The two scores are proportionally combined to render the overall score.
NUMWDMTEXTOPT	CHAR (1)	Numeric word setting. Values include N (none), A (any position), S (same position), P (any position consider punctuation), or Y (any position ignore punctuation).

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.7 MTDUPESDATA

This table contains information about duplicate records as a sample of the match results and is applicable to the Match transform. The information is used for the Match Duplicate Sample Report.

Columns	Data type definition	Description
🔑 RECNUM	INT	Sequential number assigned to each sample record output.
🔑 NAME	NVARCHAR (128)	Name of the field listed.
VALUE	NVARCHAR (256)	Value in the field listed.

> ℹ Note
>
> The OBJECT_KEY column is included in this table but it is not listed. It is a primary key common to most tables. The OBJECT_ID column is not in this table.

Related Information

Repository tables common columns [page 1745]

3.9.5.8 MTGSSRCBYSRCSTS

This table contains the inter-match and intra-match source counts for the *Source by Source Statistics* section of the Match Source Stats Summary report. This table is applicable to the Match transform.

Column	Data type definition	Description
🔑 SRCID	INT	Sequential number that identifies a source.
🔑 PROCID	INT	Sequential number that identifies a match level or an association.
GSNAME	NVARCHAR (15)	Group statistics name.
SRCNAME	NVARCHAR (256)	Source name.
OTHERSRCNAME	NVARCHAR (256)	Other source name.
TOTDUPES	INT	Total matches between the source and the other source.

> ℹ Note
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.9 MTGSSRCSTS

This table contains information about the distribution of the duplicates found in various input source records and is applicable to the Match transform. It includes information about how the duplicates were distributed as master records and subordinate records. The information is used for the Match Source Statistics Report.

Column	Data type definition	Description
🔑 SRCID	INT	Sequential number that identifies a source.
🔑 PROCID	INT	Sequential number that identifies a match level or an association.
GSNAME	NVARCHAR (15)	Group statistics name.
SRCNAME	NVARCHAR (256)	Source name.
SSSUB	INT	Single source subordinate record count for this source.
MSSUB	INT	Multiple source subordinate record count for this source.
SSMASTS	INT	Single source master record count for this source.
MSMASTS	INT	Multiple source master record count for this source.
NUMRECS	INT	Record count for this source.

> ℹ Note
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.10 MTINFO

This table contains the match set name and is applicable to the Match transform. The information is used in all of the Match reports except the Best Record Summary report.

Column	Data type definition	Description
MTSET	NVARCHAR (256)	Name of the match set.

> **Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.11 MTINSRCBYSRC

This table contains information about how often each input source matched the other input sources, and these statistics are generated by the Match transform. This table is populated once per input source group statistics object if statistics are enabled. This table is similar to the MTGSSRCBYSRCSTS table. To obtain the numbers for the source groups, the rows with the same INSRCGRPID and the same INSRCGRPID_OTHER must be added.

Column	Data type definition	Description
🔑 PROCID	INT	Sequential number used to identify a match level or an association.
🔑 INSRCID	INT	Sequential number used to identify an input source.
🔑 INSRCID_OTHER	INT	Sequential number used to identify an input source that had one or more records that matched one or more records in INSRCID.
🔑 INSRCGRPID	INT	Sequential number used to identify an input source group. A value of zero implies records that do not belong to an input source group.
🔑 INSRCGRPID_OTHER	INT	Sequential number used to identify the input source group of INSRCID_OTHER. A value of zero implies that INSRCID_OTHER does not belong to an input source group.
🔑 OBJNAME	NVARCHAR (15)	Name of the input source group statistics object.
MATCHES	INT	Sequential number used to identify the input source group of INSRCID. A value of zero indicates that INSRCID does not belong to an input source group.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.12 MTINSRCGRPINFO

This table contains input source group information and is applicable to the Match transform. It is populated once per transform when statistics are enabled and the source groups section of the input source object is defined.

Column	Data type definition	Description
🔑 INSRCGRPID	INT	Sequential number that identifies an input source group.
INSRCGRPNAME	NVARCHAR (255)	Name of an input source group.
MATCHED_COUNT	INT	Number of input records that belong to this input source group because they matched a specified input source.
DEFAULT_COUNT	INT	Number of input records that belong to this input source group because it is the default input source group.

If an input source exists that does not belong to a super source, that input source is assigned a super source ID of zero. A corresponding entry is made into the MTINSRCGRPINFO table in the following fields:

- INSPRSRCID
- INSPRSRCNAME
- MATCHED_COUNT
- DEFAULT_COUNT

All of these fields are set to zero. This allows the report code to successfully perform join operations.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.13 MTINSRCINFO

This table contains input source information and is applicable to the Match transform. It is populated once per transform if statistics are enabled and the input source object is defined.

Column	Data type definition	Description
🔑 INSRCID	INT	Sequential number used to identify the input source.
🔑 INSRCGRPID	INT	Sequential number used to identify the input source group. The number is zero if this input source does not belong to a source group.
INSRCNAME	NVARCHAR (255)	Name of the input source.
INSRCVALUE	NVARCHAR (255)	Value of the input source.
INSRCTYPE	NVARCHAR (8)	Input source type. Valid values are NORMAL, SUPPRESS, and SPECIAL.
MATCHED_COUNT	INT	Number of input records that belong to this input source because their input source value matched the value of this source.
DEFAULT_COUNT	INT	Number of records that belong to this input source because it is the defaut input source.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.14 MTINSRCMSRC

This table contains the multi-source statistics of each input source and each source group and is applicable to the Match transform. The table is populated once per input source group statistics object if statistics are enabled. To get the numbers for the source groups, the rows with the same INSRCGRPID must be added.

Column	Data type definition	Description
🔑 PROCID	INT	Sequential number that identifies a match level or an association.

Column	Data type definition	Description
🔑 INSRCID	INT	Sequential number that identifies an input source.
🔑 INSRCGRPID	INT	Sequential number that identifies an input source group. When INSRCGRPID is zero, the record does not belong to a source group.
🔑 OBJNAME	NVARCHAR (15)	Name of the input source group statistics object.
SRC2	INT	Number of records in INSRCID that are masters in match groups with a source count of 2.
SRC3	INT	Number of records in INSRCID that are masters in match groups with a source count of 3.
SRC4	INT	Number of records in INSRCID that are masters in match groups with a source count of 4.
SRC5	INT	Number of records in INSRCID that are masters in match groups with a source count of 5.
SRC6	INT	Number of records in INSRCID that are masters in match groups with a source count of 6.
SRC7	INT	Number of records in INSRCID that are masters in match groups with a source count of 7.
SRC8	INT	Number of records in INSRCID that are masters in match groups with a source count of 8.
SRC9	INT	Number of records in INSRCID that are masters in match groups with a source count of 9.
SRC10	INT	Number of records in INSRCID that are masters in match groups with a source count of 10.

ℹ Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.15 MTINSRCSELECT

This table contains input source group information and is applicable to the Match transform. This table is populated once per input source select record object if statistics are enabled. To obtain the numbers for the source groups, the rows with the same INSRCGRPID must be added. Refer to the *Reference Guide, Output flag selection options* section for more information.

Column	Data type definition	Description
🔑 PROCID	INT	Sequential number that identifies the match level or an association.
🔑 INSRCID	INT	Sequential number that identifies an input source.
🔑 INSRCGRPID	INT	Sequential number that identifies an input source group. When INSRCGRPID is zero, the record does not belong to a source group.
🔑 SECSRCID	INT	Currently not used.
🔑 OBJNAME	NVARNVARCHAR (15)	Name of the input source group statistics object.
SECSRCNAME	NVARNVARCHAR (255)	Currently not used.
SSMASTERS_SEL	NVARNVARCHAR (1)	Indicates whether the *Single source masters* option is selected (Y/N).
SSMASTERS_KEEP	INT	Number of single source master records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
SSMASTERS_DROP	INT	Number of single source master records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
MSMASTERS_SEL	NVARCHAR (1)	Indicates whether the *Multiple source masters* option is selected (Y/N).
MSMASTERS_KEEP	INT	Number of multi-source master records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
MSMASTERS_DROP	INT	Number of multi-source master records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
SSSUBS_SEL	NVARCHAR (1)	Indicates whether the *Single source subordinates* option is selected (Y/N).

Column	Data type definition	Description
SSSUBS_KEEP	INT	Number of single source subordinate records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
SSSUBS_DROP	INT	Number of single source subordinate records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
MSSUBS_SEL	NVARCHAR (1)	Indicates whether the *Multiple source subordinates* option is selected (Y/N).
MSSUBS_KEEP	INT	Number of multi-source subordinate records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
MSSUBS_DROP	INT	Number of multi-source subordinate records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
SUPSUBS_SEL	NVARCHAR (1)	Indicates whether the *Suppression subordinates* option is selected (Y/N).
SUPSUBS_KEEP	INT	Number of suppress subordinate records in INSRCID that are flagged for selection.
SUPSUBS_DROP	INT	Number of suppress subordinate records in INSRCID that are not flagged for selection.
UNIQUES_SEL	NVARCHAR (1)	Indicates whether the *Unique* option is selected (Y/N).
UNIQUES_KEEP	INT	Number of unique records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
UNIQUES_DROP	INT	Number of unique records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
SUPMASTERS_SEL	NVARCHAR (1)	Indicates whether the *Suppression masters* option is selected (Y/N).
SUPMASTERS_KEEP	INT	Number of suppress master records in INSRCID that are flagged for selection.

Column	Data type definition	Description
SUPMASTERS_DROP	INT	Number of suppress master records in INSRCID that are not flagged for selection.
SUPMATCHES_SEL	NVARCHAR (1)	Indicates whether the *Suppression matches* (normal and special records that follow a suppress record in a match group) option is selected (Y/N).
SUPMATCHES_KEEP	INT	Number of suppress match records in INSRCID that are flagged for selection.
SUPMATCHES_DROP	INT	Number of suppress match records in INSRCID that are not flagged for selection.
SUPUNIQUES_SEL	NVARCHAR (1)	Indicates whether the *Suppression uniques* option is selected (Y/N).
SUPUNIQUES_KEEP	INT	Number of unique suppress records in INSRCID that are flagged for selection.
SUPUNIQUES_DROP	INT	Number of unique suppress records in INSRCID that are not flagged for selection.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.16 MTINSRCSTATS

This table contains the statistics of each input source and each source group and is applicable to the Match transform. It is populated once per input source group statistics object if statistics are enabled.

Note

To obtain the numbers for the source groups, the rows with the same INSRCGRPID must be added.

Column	Data type definition	Description
🔑 PROCID	INT	Sequential number that identifies a match level or an association.

Column	Data type definition	Description
🔑 INSRCID	INT	Sequential number that identifies an input source.
🔑 INSRCGRPID	INT	Sequential number that identifies an input source group. When the INSRCGRPID value is zero, the record does not belong to a source group.
🔑 OBJNAME	NVARCHAR (15)	Name of the input source group statistics object.
SSMASTERS	INT	Number of records in INSRCID that are single source masters. Depending on the input source type, these could be normal or special records.
MSMASTERS	INT	Number of records in INSRCID that are multi source masters. Depending on the input source type, these could be normal or special records.
SSSUBS	INT	Number of records in INSRCID that are single source subordinates. Depending on the input source type, these could be normal or special records.
MSSUBS	INT	Number of records in INSRCID that are multi source subordinates. Depending on the input source type, these could be normal or special records.
SUPSUBS	INT	Number of records in INSRCID that are suppress subordinates.
UNIQUES	INT	Number of records in INSRCID that are unique. Depending on the input source type, these could be normal or special records.
SUPMASTERS	INT	Number of records in INSRCID that are suppress master records.
SUPMATCHES	INT	Number of records in INSRCID that are suppress matches (normal and special records that follow a suppress record in a match group).
SUPUNIQUES	INT	Number of suppress records in INSRCID that are unique.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.17 MTKEYDEF

This table contains match criteria information for each match set and is applicable to the Match transform. The information is used in the Match Criteria Summary report.

Column	Data type definition	Description
🔑 KEYID	INT	Number that represents the field type of the key field.
🔑 KEYFLDNUM	INT	Number that indicates which occurrence of the key field this is. The first occurrence will have a value of 1, the second will have a value of 2, and so on. An example of a key field that supports multiple occurrences is the Given Name field, where each occurrence is a different person.
🔑 KEYFLDALTNUM	INT	Number that indicates which alternate (match standard) of the key field this is. The original data will have a value of 0, the first alternate will have a value of 1, the second alternate will have a value of 2, and so on.
KEYNAME	NVARCHAR (256)	Name assigned to the key field when it was created in the job.
KEYLEN	INT	Number of characters in the database field that are used in creating the key field.
DATARECFLDNAME	NVARCHAR (256)	Field's input mapped name.
RMVPUNCT	CHAR (1)	Indicates whether the field was preprocessed by removing punctuation (Y/N).
CONVTOUPPER	CHAR (1)	Indicates whether the field was preprocessed by converting text to upper case (Y/N).
STDDIACHRS	CHAR (1)	Indicates whether the field was preprocessed by converting diacritical characters (Y/N).
CONVTXTTONUM	CHAR (1)	Indicates whether the field was preprocessed by converting numbers represented by text to numerals (Y/N).

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.18 MTPROCESS

This table contains information about the Match transform processing and is applicable to the Match transform. The information is used in all of the Match reports except the Best Record Summary report.

Column	Data type definition	Description
🔑 PROCID	INT	Sequential number that identifies a match level or an association.
PROCTYPE	CHAR (1)	Process type. Valid values are B (Break Group Process), M (Match Level Process), or A (Association Process).
PROCNAME	NVARCHAR (15)	Process name.

Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.9.5.19 MTRULESRES

This table contains information about the effect of the criteria on the total matching process and is applicable to the Match transform. The information is used in the Match Contribution Report (Criteria Information subreport).

Columns	Data type definition	Description
🔑 PROCID	INT	Sequential number that identifies a match level or an association.
🔑 CMPCRITID	INT	Sequential number that identifies the criteria of a match level.
🔑 CRITID	INT	Sequential number that identifies an individual criteria. A value of 999 is used for the weighted results.
MTDECS	INT	Number of match decisions made by this criteria.
NOMTDECS	INT	Number of no match decisions made by this criteria.
CTDECS	INT	Number of records that were allowed to continue with the match process after a no match decision was made.

i Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1745]

3.10 Locales and Multi-byte Functionality

Data Services supports the use of different locales in sources, the Job Server, and targets. It also supports single and multi-byte code pages. By combining these settings, you can control processing across different languages and allow for differences in capitalization, time and date formats, and character sets.

Related Information

Definitions [page 1808]
Supported locales and encodings [page 1810]

3.10.1 Language packs

Language packs are available for installation, and provide you with locales, other than English (the default locale), for viewing the Data Services user interface and any text that the user interface generaties in other languages.

There is no need to reinstall Data Services to acquire a language pack; they can be installed over an existing Data Services installation.

After a language pack is installed, you will be able to select the locale for both the user interface and the displayed data.

- *Product locale:* Specifies the user interface language and all product messages.
- *Preferred viewing locale:* Specifies the locale that the user data should be presented in. For example, date formatting should be presented in the preferred viewing locale.

There are two locations for setting these options: the Locale Selector and the *Tools* > *Options* window in the Designer.

Where you make your selections depends on your installation configuration.

3.10.1.1 To set locales in the Designer

If you are including the Designer in your installation, you can set your locales in the Designer itself.

> i Note
>
> Changing the locale settings in the Designer's *Options* window will automatically change the locale settings in the Locale Selector.

1. In the Designer, choose *Tools* > *Options* .
   The *Options* window opens.
2. Expand the *Designer* category, and select *Language*.
3. Select a value for the *Product Locale* and the *Preferred Viewing Locale* options.

3.10.1.2 To set locales in the Locale Selector

Perform this task if you are installing the engine but not the Designer.

> i Note
>
> Changing the locale settings in the Locale Selector will automatically change the locale settings in the Designer's *Options* window.

1. Access the Locale Selector.
   ○ Windows: Choose *Start* > *Programs* > *SAP Data Services* `<x.x>` > *Data Services Locale Selector*

- UNIX/Linux: From the command line, type `./start LocaleSelector.sh`

The Locale Selector window opens.

2. Select the locales for the *Product locale* and the *Viewing locale* options.
3. Select the locale for the *Server Log Locale*.
4. To specify the Language, territory, and code page to use for the repository connection and for processing data:
   - To use the default locale, select *Use default database locale*
   - Select the locales for the *Language and Territory* and *Code Page* options.

> **Note**
>
> utf-16 is supported as a code page but cannot be selected in locale selector code page dropdown list.

3.10.1.3 To set locales in UNIX or Linux

Use this procedure to modify the product locales on a UNIX or Linux system.

1. Locate and open your `DSConfig.txt` file. (The default directory is **`<$LINK_DIR>`**`/conf/`.)
2. In the `[Locales]` section, change the `ProductLocale` options to the locale you want.
3. Save and close the `DSConfig.txt` file.

3.10.1.4 Impact of locale settings on Data Services components

The locale settings you choose impact Data Services components differently. Here is a list of those impacts:

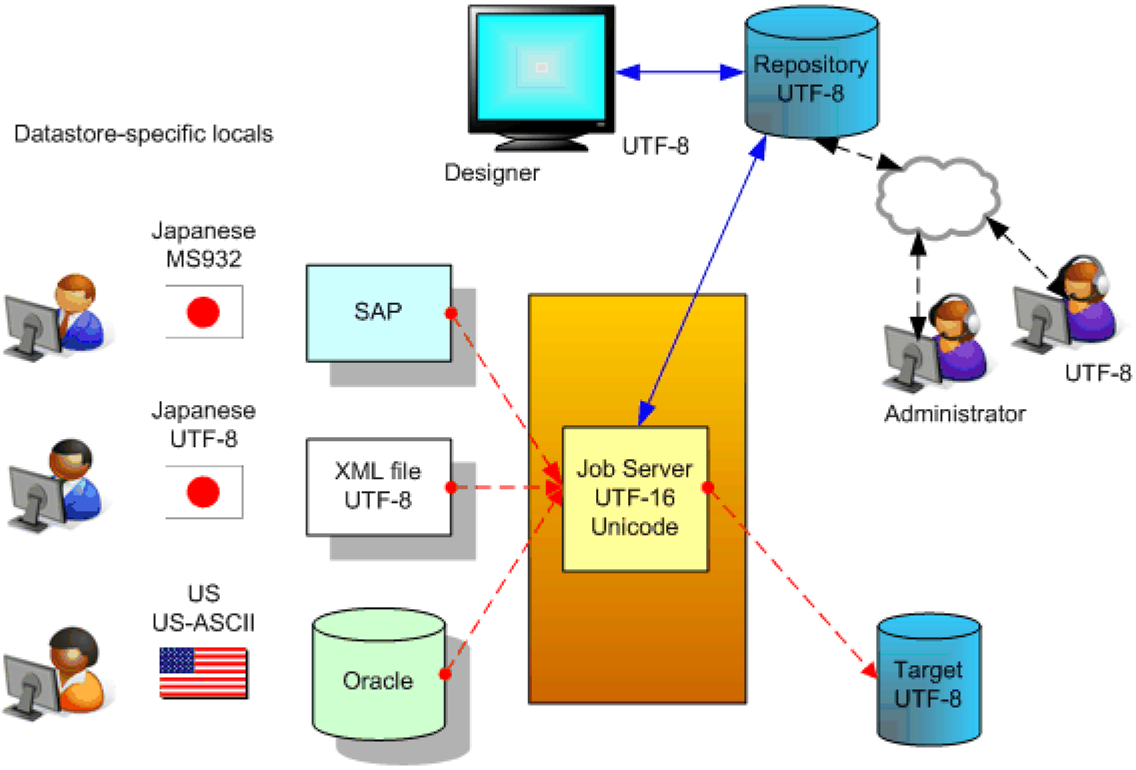
Component	Description
License Manager	License Manager always displays in English.
Management Console	The Management Console's locales are controlled in your internet browser's locale setting.
Documentation	Documentation that is accessed from the Designer will display in the same language as the Designer's *Options* window *Product locale* setting.
Log files	Messages written to a log file will be in the language set in the *Server Logs locale* option in the Locale Selector. This affects only the text of the log files. The date and timestamps remain the same.

3.10.2 Locale support

By supporting different locales Data Services allows you to configure an enterprise environment in which components process data in different human languages and then load the data to a target data code page.

For example, you can configure locales for the following sources:

Language	Territory	Code page
Japanese	Japan	shift_jis
Japanese	Japan	UTF-8
English	United States	US-ASCII



To use Data Services' locale support, set locales within each of the following:

- Database (source, target, and repository) or application (SAP, SAP Master Data Services, PeopleSoft, Oracle, Siebel, JDE).
- Database client.
- Datastore connection to a database or application.
  Datastore locales must match the locales of source and target database clients. This allows the datastore to move data between Data Services and each database without possible data corruption. If the database locale differs from its database client locale, it is your responsibility to ensure that the database transcodes the data before it reaches or after it leaves Data Services.

i Note

To avoid the necessity of setting locales for a database client and a Data Services datastore, you can process with or without UTF-16 Unicode.

i Note

All adapter datastores are automatically set by Data Services to the code page UTF-8. They are handled the same way as XML message sources and targets.

- File format (flat file, XML Schema, and DTD): Match the file format locale to that of each source or target file.
- Job Server: Data Services uses the Job Server's locale for the engines it spawns. The Designer also uses the Job Server's locale as its repository connection locale. To avoid possible data corruption, in case the SAP Sybase repository or DB2 repository database codepage is not UTF-8, use the same locale settings for the repository, its client, and the Job Server.

Related Information

Processing with and without UTF-16 Unicode [page 1799]

File format locales [page 1803]

3.10.2.1 Locale selection

Data Services automatically sets the locale for the following components:

Job Server

After installation, the locale of the Job Server is set to `<default>` which enables Data Services to automatically set the locale for the repository connection (for the Designer) and to process job data (for the Job Server) according to the locale of the datastore or operating system. This capability enables Data Services to automatically change the locale for better performance (for example, set the locale to non-UTF-8 if the datastore is non-unicode data).

The following table shows different datastores and Job Server locale settings and the locale that Data Services automatically sets for the data flow. In this table, the Job Server locale is set to `<default>` and derives its value from the operating system.

Datastore 1 locale	Datastore 2 locale	Job Server locale	Data flow locale
Single-byte code page	Multi-byte code page	Single-byte or Multi-byte code page	Unicode
Multi-byte code page	Multi-byte code page	Single-byte code page	Unicode
Multi-byte code page	Multi-byte code page	Multi-byte code page	Unicode

Datastore 1 locale	Datastore 2 locale	Job Server locale	Data flow locale
Single-byte code page 1	Single-byte code page 2	Single-byte code page 3	Unicode
Single-byte code page 1	Single-byte code page 2	Multi-byte code page	Unicode
Single-byte code page 3	Single-byte code page 3	Multi-byte code page	Unicode
Single-byte code page 3	Single-byte code page 3	Single-byte code page 1	Single-byte code page 3

The following table summarizes the locale that Data Services now sets for each data flow when the locale of the Job Server is set to `<default>`. Different data flows in the same job can run in either single-byte or Unicode.

Locale of datastores in data flow	Job Server locale	Locale that Data Services sets
One datastore has multi-byte locale	Single-byte or multi-byte	Unicode
Different single-byte locales	Single-byte or multi-byte	Unicode
Same single-byte locale	Multi-byte	Unicode
Same single-byte locale	Single-byte	Single-byte

Designer

The Designer uses the Job Server locale to move data between the Designer and the repository.

The Designer expects to receive data from the repository in the Job Server's locale. Objects you create in the Designer are represented internally using a textural language (ATL) that is sent to and received from the repository's database client in the form of SQL statements. Use the same locale when installing the Job Server as you set for your repository and its client to support Data Services' internal language.

The Designer also has its own locale which is automatically set to that of its Microsoft Windows operating system locale. The Designer automatically transcodes input data from its locale to the Job Server locale when it interacts with the repository.

Management Console

The Management Console's locale is automatically set to UTF-8. By using UTF-8 (a Unicode encoding that supports all languages), Data Services ensures data integrity in the Management Console. All Data Services logs (error, trace, and monitor) are generated by the engine in UTF-8. When the Designer reads logs, it transcodes their content from UTF-8 to the Designer locale.

3.10.2.1.1 To override the default Job Server locale

You can override the default locale for the Job Server by using the Data Services Locale Selector utility.

Choose *Start* > *Programs* > *SAP Data Services* **`<x.x>`** > *Data Services Locale Selector*.

For more information, see “Guidelines for setting locales”.

3.10.2.2 Code page support

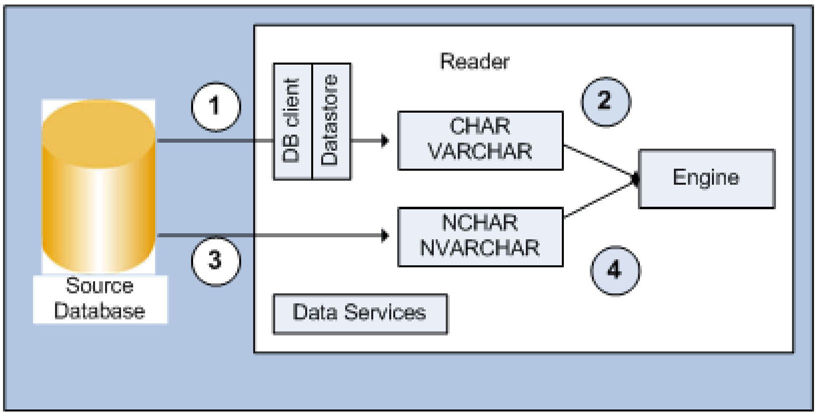
The code page you select in each locale used by a job determines whether transcoding occurs during job processing. Data Services automatically transcodes between different code pages when necessary to support complex, multi-language data management in a single job.

However, it is your responsibility to set up the connections between Data Services and its repository, sources, and targets to avoid data corruption due to mismatched locale settings between Data Services and its connections to external systems. Therefore, ensure you use the same code page as the database middleware to which you are connecting.

3.10.2.2.1 Processing with and without UTF-16 Unicode

The Job Server supports reading and loading national character-set data types (`nchar`, `nvarchar`, `nvarchar2`, `graphic` and `var-graphic`). Data Services automatically handles these data types using UTF-16. If extracting or loading data that uses only these data types, you do not have to set locales for a database client and its datastore. If the data is multi-byte, the Job Server locale should be set to a multi-byte code page, such as `UTF-8` or `shift_jis`.

For example, here is the path that your data takes from a source database to a Data Services engine when data is read during a job run.



The upper arrow shows a normal job run and the locales the job uses to support data integrity.

1. Data Services assumes the database transcodes data into the database client code page, as needed, before it uses its datastore code page to read data from the database. The client and datastore code pages must match otherwise Data Services cannot recognize the format of the data. Note that the data types are not national character-set data types.
2. If the datastore code page is different from the Job Server code page, Data Services transcodes it then processes the job. Note that the client and datastore are not in the path.
   The lower arrow shows a national character-set data type job run:
3. Data Services reads the data formatted for national character-set data types using a UTF-16 code page without using client and datastore code pages.
4. The engine transcodes the data from UTF-16 to the Job Server's code page before it is processed.

When data continues to a target database, the processes are reversed. Data Services automatically transcodes the data in the national character-set data type path back into UTF-16 before loading it into a target. In a normal job run path, if the datastore code page differs from the Job Server code page, the engine transcodes the data to the target's code page before it is passed on to the database client. The datastore locale and the database client locale must match so that data is accurately sent into the database where it might again need to be transcoded into a different locale for storage.

Data Services support of national character-set data types is restricted to the specific source and target databases. For example:

- Oracle with `nchar` or `nvarchar2`
- Microsoft SQL Server with `nchar` or `nvarchar`
- DB2 with `graphic` or `vargraphic`

Data Services can also extract, transform, and load a single table with both national character-set data types and other data types. In this case, the data in the columns with the national character-set data types uses the UTF-16 path and the other data uses the datastore path through Data Services.

In the Designer, you can also assign Unicode as a code page by creating a Microsoft SQL Server datastore connection and selecting *UTF16* or *UTF8* as its code page.

National character-set data types help you avoid having to set locales for connections to database and application sources and targets. However, you still must set locales for the Job Server and for file formats (if you use files in your job).

Also, when Data Services imports a table with columns using any character data type (`nchar`, `nvarchar`, `varchar, char`, and so on), it imports the column size in number of characters (not bytes). Similarly, while creating a new column in Query objects, Data Services assumes the column size is in number of characters. As the number of bytes per character varies from code page to code page, at runtime, the Data Services engine allocates memory based on the Job Server's code page.

Related Information

Column sizing [page 1808]

3.10.2.2.2 Minimizing transcoding

As a rule, transcoding impacts job performance.

- Use the same locale for all components and use a single-byte code page if possible.
- If a datastore or file format and the Job Server use a different locale, Data Services automatically transcodes the data, which supports a multi-language enterprise environment.

Data Services minimizes the impact of transcoding for equivalent code sets such that transcoding between the following code page pairs does not impact performance.

Superset	Subset
cp1252	ISO88591 (LATIN1)
cp1250	ISO88592
cp1251	ISO88595
cp1253	ISO88597
cp1254	ISO88599
cp1255	ISO88598
cp1256	ISO88596
cp1257	ISO88594

Related Information

Supported locales and encodings [page 1810]

3.10.2.3 Guidelines for setting locales

3.10.2.3.1 Job Server locale

The Job Server locale is used by the engines it spawns as well as the Designer's repository connection locale.

If the locale of the Job Server is set to `<default>` after installation, the Job Server takes its locale from the operating system of the host computer where it is installed. If you process multi-byte data with the operating system locale set to single-byte, set the Job Server's locale to the same code page or a superset of the data code page to avoid data corruption. For more information, see Example 3 in the “Example locale settings” section.

If your jobs will run in a multi-language environment, Data Services automatically sets the Job Server's locale to a superset of all datastore and file format locales.

If you do not have a multi-language environment, use a single-byte code page and use the same settings for all locale values or use only locales with code pages that minimize transcoding. This strategy ensures the best performance.

You can override the default locale for the Job Server by using the Data Services Locale Selector utility. Choose *Start* > *Programs* > *SAP Data Services* **<x.x>** > *Data Services Locale Selector*.

Also make sure that the locale of the repository's database client (installed on the Designer's computer) matches the Job Server locale. The Designer uses the Job Server's locale to ensure that it passes accurate data to the repository.

Related Information

Minimizing transcoding [page 1801]
Supported locales and encodings [page 1810]
Example locale settings [page 1805]

3.10.2.3.2 Database, database client, and datastore locales

Set a database and its database client locales using your database software.

Set a Data Services datastore to the same locale as the application or database client to which it connects. Data Services automatically sets each datastore locale to `<default>` in order to match that of the Job Server. However, if your sources or targets use different locales, manually modify the *Language* and *Code page* options under the *Advanced* button in the Datastore Editor.

When you view table data with the View Data feature of the Designer, Data Services formats the values of numeric data type columns according to the number format of its locale territory. For example, suppose a datastore that is connected to an Oracle database with a datastore locale of `deu_de.cp1252`. With the Data Services locale set to `eng_us.cp1252`, View Data will display numeric values with a dot (.) as the decimal separator.

When reading and loading numeric data from databases, Data Services automatically determines the number format appropriate for each database, which does not depend on the Job Server locale territory. However, if the datastore table contains numeric values in string data type columns and an implicit conversion from string to numeric data type is required, Data Services expects that the number format matches the format of its locale territory.

If the format of numeric values in string data type columns does not match the Job Server locale territory, use the `to_decimal_ext` function or the `to_decimal` function to convert the string to a numeric data type by specifying the correct thousand and decimal separators. Similarly, when loading numeric values to string-type columns in a datastore, Data Services formats numbers according to the Job Server locale territory format. If you need to convert the data to a number format used by a different territory, use the `to_char` function.

Locales apply for all profiles created from each datastore.

All adapter datastores are automatically set by Data Services to the code page UTF-8. They are handled the same way as XML message sources and targets.

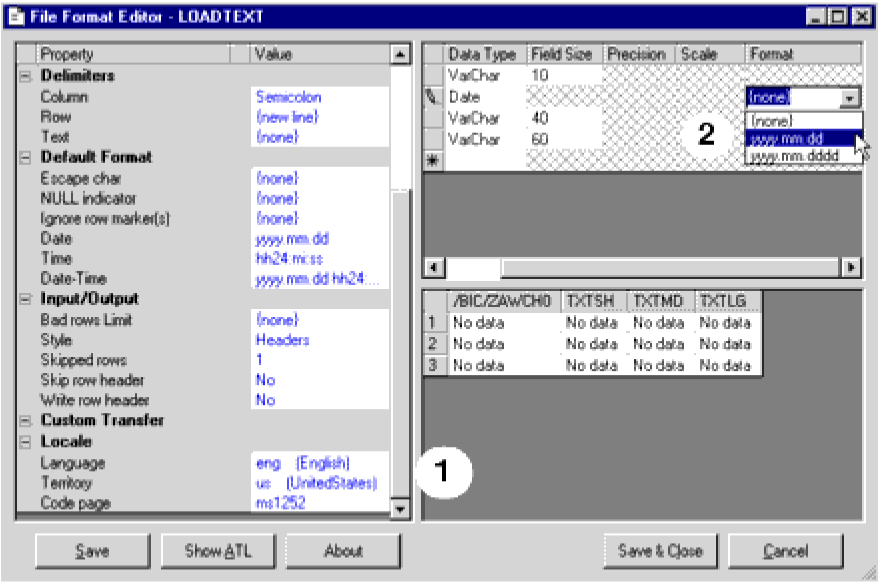
Related Information

Datastore [page 860]
XML encodings [page 1804]
to_char [page 1676]
to_decimal [page 1679]
to_decimal_ext [page 1680]

3.10.2.3.3 File format locales

For flat files, the `<default>` locale is automatically set to match that of the Job Server. However, you can use the *Locale* section on the file format editor to specify the language and code page that corresponds to the file data.

Refer to the following image:



File Format Editor - LOADTEXT

Property	Value
Delimiters	
Column	Semicolon
Row	{new line}
Text	{none}
Default Format	
Escape char	{none}
NULL indicator	{none}
Ignore row marker(s)	{none}
Date	yyyy.mm.dd
Time	hh24:mi:ss
Date-Time	yyyy.mm.dd hh24:...
Input/Output	
Bad rows Limit	{none}
Style	Headers
Skipped rows	1
Skip row header	No
Write row header	No
Custom Transfer	
Locale	
Language	eng {English}
Territory	us {UnitedStates}
Code page	ms1252

Data Type	Field Size	Precision	Scale	Format
VarChar	10			
Date				{none}
VarChar	40			
VarChar	60			

	/BIC/ZAWCH0	TXTSH	TXTMD	TXTLG
1	No data	No data	No data	No data
2	No data	No data	No data	No data
3	No data	No data	No data	No data

1. As in datastore locales, the *Territory* option for a flat file's locale is not active in Data Services.
2. However, you can change the data type for a flat file column and then enter the format that you want to use for a `date` or any numeric data type using the *Column Attributes* work area.

Related Information

File format [page 920]
Designer Guide: File Formats, Number formats [page 264]

3.10.2.3.4 XML encodings

For XML Schemas or DTDs used as sources, the default encoding (code page) for messages is assumed by Data Services to be UTF-8. Header information is ignored. Data Services transcodes inbound data to the Job Server code page (if necessary) before processing it. For XML files Data Services reads header information at run-time. If encoding information is not specified, Data Services assumes it is UTF-8. To edit the encoding for XML file sources, do so in the original file then rerun the job.

For XML Schemas or DTDs used as targets, the default encoding is automatically set to UTF-8.

- If you use an XML Schema or DTD as a file, you can change the encoding using the target editor.
- If you use an XML Schema or DTD as a message, the encoding cannot be changed. Data Services transcodes out-bound message configuration information (like global variable data) into UTF-8.

Related Information

Target XML files, messages, and templates [page 994]

3.10.2.3.5 Overriding the database client locale

By default, most database clients set their locale to the same locale used by the operating system. However, you can override the default setting by using either an environment variable or client configuration tool. We recommend using the methods below to override the client locale for different database types.

Database	Default setting	Environment variable	Driver setting
DB2	OS Locale	DB2CODEPAGE	N/A
Informix	OS Locale	CLIENT_LOCALE	Client Locale field
MySQL	Latin1	N/A	Character Set field
Netezza	N/A	N/A	ASCII (8-bit driver) UTF16 (Unicode driver)
Oracle	OS Locale	NLS_LANG	NLS_LANG
SAP Sybase ASE	OS Locale	LC_ALL	N/A
SAP Sybase IQ	OS Locale	N/A	Character Set field
Teradata	ASCII	N/A	Session Character Set field

> i Note
>
> For Microsoft SQL Server and NeoView databases, the only way to change the database client locale is to change the operating system locale. For generic ODBC databases, see your ODBC driver configuration documentation.

3.10.2.3.6 Example locale settings

Example 1

Suppose you are running Data Services on a Windows system that is set to Japanese and want to process Japanese data, with the Data Services code page set to default, `<default>_<default>.<default>`. All datastore and file formats are also set to the `<default>` code page.

In this situation, the Job Server takes its locale from the server's system locale, `jpn_jp.shift_jis`. All datastore and file formats get the `shift_jis` codepage.

To access the data correctly, the database client must also be configured for a locale equivalent to `jpn_jp.shift_jis`. Refer to the database client documentation for information on how to obtain and configure the correct locale name.

Example 2

Suppose you are running Data Services on a Windows system and want to extract data from multiple databases using different character sets, such as `shift_jis`, `latin1`, and `latin9`, and then load that data to a single UTF-8 target database.

To support this scenario, set the Data Services code page to `eng_US.utf8`, and configure each datastore locale to match the locale of the linked database.

⚠ Restriction

Because the database client code page must match the database server code page, there can be only one client character set per system. This restriction does not apply to databases that allow you to configure the character set via an ODBC driver setting.

Example 3

Suppose you are running Data Services on a Windows system that is set to English (`ms1252`) and want to process Japanese data with the datastore code page set to the `<default>` code page. You must set the Job Server's code page to Japanese (`shift_jis`) to avoid data corruption.

3.10.3 Multi-byte support

Data Services supports various multi-byte code pages that are typically specific to each language. Data Services also provides support for two Unicode encodings: UTF-8 and UTF-16. These are multi-byte code pages that support most of the world's languages.

3.10.3.1 Multi-byte string functions

All Data Services string functions support multi-byte data.

For example, when using the functions INDEX, LENGTH, RPAD, and SUBSTR, the sizes and offsets used as arguments or return values are expressed as number of characters, not number of bytes. You can also see this logical, intuitive behavior when these functions are pushed down and evaluated by the database. Similarly, when Data Services evaluates the SQL predicate `LIKE`, the single-character wildcard "_" matches exactly one character, not one byte.

3.10.3.2 Numeric data types: assigning constant values

Use care when assigning a constant to a numeric variable or column in Data Services.

Assigning a value as a numeric directly

If a numeric value is not within quotes—for example, `$comm = 123.45`—then Data Services stores the value as a numeric. If the value is stored as numeric, then dot (".") is the only recognized decimal separator in the Designer, regardless of the locale. While executing the job, however, the Data Services engine automatically converts the value to appropriate decimal separator for the locale.

Syntax	Comment
`$VALUE = 123.45`	Correct syntax.
`$VALUE = 123,45`	Wrong syntax. Use a comma "," as a decimal separator inside Data Services only if the opertating system's decimal separator is a comma "," and the value is within quotes.

Assigning a value in string format

If a numeric value is stored within quotes—for example, `$comm = '123,45'`—then the Designer stores the value in a string format and while executing the job, the Data Services engine automatically converts the value from the string to the appropriate numeric data type. When the Data Services engine converts a string to a numeric format, it uses the decimal separator for the specified Job Server locale.

For example, if the Job Server locale is set to German regional settings (`deu_de.cp1252`), then Data Services uses a comma (",") as the decimal separator when converting a value from a string to appropriate numeric data type.

Syntax	Comment
`$VALUE = '123,45'`	Correct syntax.

Syntax	Comment
`$VALUE = '123.45'`	Wrong syntax. If the operating system's decimal separator is a comma "," and the value is within quotes, then Data Services tries to process a dot "." as a thousands separator.

If the Job Server locale is set to English regional settings (`eng_us.cp1252`), then Data Services uses a dot (".") as the decimal separator when converting a value from a string to an appropriate numeric data type.

Syntax	Comment
`$VALUE = '123.45'`	Correct syntax.
`$VALUE = '123,45'`	Wrong syntax. If the operating system's decimal separator is a dot "." and the value is within quotes, then Data Services tries to process a comma "," as a thousands separator.

i Note

Using the incorrect decimal separator can generate incorrect results. Similarly, having a thousands separator can generate incorrect results. It is recommended that you do not use a thousands separators when converting a value from string to numeric.

3.10.3.3 Byte Order Mark characters

The Unicode standard includes the use of Byte Order Mark (BOM) characters as a signature for file data.

Data Services supports BOM characters. When it reads data from a file, the Data Services engine trims BOM characters. Data Services supports the following BOM characters and their corresponding encodings:

BOM Characters (Bytes)	Encoding Form
`FE FF`	UTF-16 big-endian
`FF FE`	UTF-16 little-endian
`EF BB BF`	UTF-8

3.10.3.4 Round-trip conversion

While transcoding, if Data Services encounters round-trip conversion conflicts, it transcodes to the first code point match (ascending order of the hexadecimal values) in the target code page. For example, consider two different Japanese characters that are mapped to different hexadecimal code points in the shift_jis code page (`EEFA` and `FFE4`) but then are mapped to the same UTF-16 code point (`FFE4`). In this case, when transcoding back from UTF-16 to shift_jis, Data Services converts them both to code point `EEFA`.

3.10.3.5 Column sizing

The number of bytes per character can vary from one code page to another. For example, the "£Á" which represents a special "A" in the Korean ksc-5601 code page, needs 2 bytes to represent the character, while the UTF-8 code page needs 3 bytes to represent the same character.

Code page	Value	Hex values	Bytes
ksc-5601	special A	`A3 C1`	2
UTF-8	special A	`EF BC A1`	3

Data Services represents sizes in number of characters. Internally, Data Services allocates enough memory to store multi-byte characters.

If the datastore code page is different from the Job Server code page, then transcoding must occur, which may result in the need for extra space allocation.

3.10.4 Limitations of multi-byte support

There are several limitations to SAP Data Services' support for multi-byte characters:

- The software supports a variety of single- and multi-byte code pages, but it does not support UCS-4, or SAP application blended code pages. In addition, the software does not support Surrogate Pairs. EBCIDIC is supported for COBOL files only.
  The software does not support EBCDIC code pages for datastores as they are not ASCII- or UTF-8 -compatible. This is not a problem for users targeting or sourcing data from IBM systems if the engine is running on a non-IBM platform. The IBM system will transparently convert its data to/from EBCDIC when communicating with a foreign architecture.
- Each data flow can process in different locales. However, if you need to change the default locale setting, you need to use a different installation of the Job Server on a different computer, because the automatic locale setting will be disabled.
- The software does not yet fully address all formatting issues. For instance, it supports the "dot" and "comma" currency formats used for most European currencies, but does not support the "tick" and "space" currency format used in Switzerland.

3.10.5 Definitions

With regard to locales and multi-byte functionality associated with Data Services, refer to the following terms:

Locale

Consists of three values related to world regions that control the format of data when it is stored, processed, or displayed. To specify a locale for the Job Server, you must select a Language, Territory, and Code page value.

Datastore and file format locales do not require that you set the Territory value for a locale.

Database sources and targets might not need locale values specified.

Language

Specifies the locale value for a human language.

Territory

Specifies the locale value for a geographical location (usually the country) where a locale language is used. The paring of a language with a territory determines factors such as date format, time format, decimal separator, currency format, and so on. Data Services uses territory values to process the following data types:

- date
- datetime
- decimal
- double
- int
- interval
- numeric
- real
- time
- timestamp

Code page

A table of characters that associates each character with a numeric index (code point value). Data Services uses a code page value to transcode varchar data types. Most languages require their own code pages, although some code pages can represent multiple languages. Most code pages are compatible with `US-ASCII` for code points below 128.

This means for example, that the Japanese code page shift_jis also partially supports other languages such as English. However, use `Japanese` as the Language value, `Japan` as the Territory value, and `shift_jis` as the Code page to avoid possible data corruption in a Japanese locale. It is your responsibility to select corresponding values for locales. While Data Services validates that locale values are entered, it does not validate that they are realistic.

Single-byte

An encoding or code page in which each character is represented by one byte.

Multi-byte

An encoding or code page in which each character is represented by one or more bytes. Some languages, like Korean and Chinese, can only be represented by multi-byte characters. Use multi-byte code pages to process characters for those languages.

Encoding

The process of representing a code page character as one byte (single-byte encoding) or a sequence of bytes (multi-byte encoding).

Transcode

Converts data from one code page to another.

To support ETL environments in which sources with different locales are processed in the same job, Data Services supports transcoding. Note that transcoding can impact performance.

Unicode

Provides a unique number for every character and a method to implement ISO/ISE 10646. Data Services supports UTF-8 and UTF-16 Unicode transformation formats.

UTF-8

Serializes a Unicode code point as a sequence of one to four bytes depending on the complexity of the character (single-byte characters use one byte and multi-byte characters use up to four). Data Services allows you to select UTF-8 as a code page for the Job Server and connections to sources and targets.

UTF-16

In Data Services, standardizes each Unicode code point at two bytes for each character. Allows access to 63k characters as 16-bit units.

Data Services supports UTF-16 for:

- A Microsoft SQL Server database when its datastore code page is set to utf8 or utf16
- National character-set data types in the following databases:
  - Oracle with nchar or nvarchar2
  - Microsoft SQL Server with nchar or nvarchar
  - DB2 with graphic or vargraphic

When using UTF-16 support, you do not have to set locales for connections to database sources and targets.

3.10.6 Supported locales and encodings

For a language, territory, code page or encoding you can also select <default>. For a Job Server locale, this means that the value is read from the operating system's locale. For example, you log in to your Windows session as Davis with a user locale of eng_gb.cp1252. When you install a Job Server, it picks up the same locale and displays it as the default (eng_gb.cp1252). At this point you can:

- Accept these values. The Job Server will always use the eng_gb.cp1252 locale.
- Use default for one or all values, for example by entering <default>_<default>.<default>. The Job Server's locale will always match the operating system's locale.
- Edit this locale to, for example eng_us.cp1252. The Job Server will always use the eng_us.cp1252 locale.

Your choice will depend on how you want to set up your system.

Note

The combination of language code zho and territory cn maps to Simplified Chinese, while the combination of zho and tw maps to Traditional Chinese.

Supported languages

SAP Data Services supports all three-letter language abbreviations specified in the ISO 639-2/T standard.

Supported territories

SAP Data Services supports all two-letter territory abbreviations specified in the ISO 3166-1 standard.

Supported code pages

SAP Data Services supports the following code pages:

Code page	Description			
	XML Encoding	Multi-byte	Vendor	Unicode Ver.
big5	Traditional Chinese Big 5 plus Microsoft extensions. User-defined range added to match Windows 2000 SP4.			
	big5	Yes	Microsoft code page 950	3.0
big5-hkscs	Traditional Chinese Big 5 plus Hong Kong Supplementary Character Set.			
	big5-hkscs	Yes	Microsoft code page 950	3.0
cp1250	Latin-2 (Central Europe)			
	WINDOWS-1250		Microsoft code page 1250	2.1
cp1251	Cyrillic (Slavic)			
	WINDOWS-1251		Microsoft code page 1251	2.1
cp1252	Latin-1 (ANSI), ISO 8859-1 plus Microsoft extensions			
	WINDOWS-1252		Microsoft code page 1252	2.1
cp1253	Greek			
	WINDOWS-1253		Microsoft code page 1253	2.1
cp1254	Latin-5 (Turkish), ISO 8859-9 plus Microsoft extensions			
	WINDOWS-1254		Microsoft code page 1254	2.1
cp1255	Hebrew			
	WINDOWS-1255		Microsoft code page 1255	2.1
cp1256	Arabic			
	WINDOWS-1256		Microsoft code page 1256	2.1
cp1257	Baltic Rim			
	WINDOWS-1257		Microsoft code page 1257	2.1
cp1258	Vietnamese			
	WINDOWS-1258		Microsoft code page 1258	2.1
cp936	Simplified Chinese, GB 2312-80 plus Microsoft extensions. User-defined range added to match Windows 2000 SP4.			
	iso2022cn	Yes	Microsoft code page 936	3.0
euc-jp	Japanese Extended UNIX Code (incl. JIS X 0212)			
	EUC-JP	Yes	Japanese EUC (JIS X 0201-1976, JIS X 0208-1990, JIS X 0212-1990)	2.1
euctw	Traditional Chinese (Taiwan) Extended UNIX Code			

Code page	Description			
	XML Encoding	Multi-byte	Vendor	Unicode Ver.
	EUC-TW	Yes		
gb18030	Chinese National Standard (supports both simplified and traditional Chinese characters)			
	GP18030	Yes		
ibm-874_ p100-1995	Thai			
		Yes		
iso-8859-1	Western European			
	ISO-8859-1		ISO/IEC 8859-1:1987	2.1
iso-8859-15	Latin-9, Western European			
	ISO-8859-15		ISO/IEC 8859-15:1999	2.1
iso-8859-2	Latin-2, Eastern European			
	ISO-8859-2		ISO/IEC 8859-2:1987	2.1
iso-8859-3	Latin-3, Southeast European			
	ISO-8859-3		ISO/IEC 8859-3:1988	2.1
iso-8859-4	Latin-4, Baltic			
	ISO-8859-4		ISO/IEC 8859-4:1988	2.1
iso-8859-5	Cyrillic			
	ISO-8859-5		ISO/IEC 8859-5:1988	2.1
iso-8859-6	Arabic			
	ISO-8859-6			
iso-8859-7	Greek			
	ISO-8859-7		ISO/IEC 8859-7:1987	2.1
iso-8859-8	Hebrew			
	ISO-8859-8		ISO/IEC 8859-8:1988	2.1
iso-8859-9	Latin-5, Turkish			
	ISO-8859-9		ISO/IEC 8859-9:1989	2.1
ksc-5601	Korean KS C 5601-1992 plus Microsoft extensions. Currency mapping changed and user-defined range added to match Windows 2000 SP4.			
	KSC_5601	Yes	Microsoft code page 949	3.0
shift_jis	"Standard" Japanese Shift-JIS without Microsoft extensions			
	Shift_JIS	Yes	Shift-JIS (JIS X 0201-1976, JIS X 0208-1990)	2.1
us-ascii	7-bit ASCII			

Code page	Description			
	XML Encoding	Multi-byte	Vendor	Unicode Ver.
	ISO-8859-1		ISO/IEC 646	
utf-16	UTF-16 encoding of Unicode			
	UTF-16	Yes		2.1
utf-16be	UTF-16be (big endian) encoding of Unicode			
	UTF-16be	Yes		2.1
utf-16le	UTF-16le (little endian) encoding of Unicode			
	UTF-16le	Yes		2.1
utf-8	UTF-8 encoding of Unicode			
	UTF-8	Yes		2.1

3.11 Python

3.11.1 Python

SAP Data Services supports the Python programming language for writing expressions with the User-Defined and Match transforms. In this section, you can find explanations of Python methods and some examples.

If you want more information about the Python language, see the Python help file, which is installed with the software. By default, it is located at **`<LINK_DIR>`**`\DataQuality\python\doc\python25.chm`. The Python web site, at *www.python.org* , also contains valuable information.

Related Information

About Python [page 1814]
Create an expression with the Python Expression editor [page 1817]
Defined classes and methods [page 1821]
FlDataCollection class [page 1822]
FlDataManager class [page 1827]
FlDataRecord class [page 1830]
FlProperties class [page 1832]
FlPythonString class [page 1834]
Python examples [page 1836]

3.11.1.1 About Python

Python is an open-source, object-oriented scripting language. Python is installed with your SAP Data Services installation; therefore, you are equipped with everything you need to begin coding. The software even has its own Python Expression editor, which is accessed from the Match and User-Defined transforms.

By using Python with the software, you can customize transforms to meet your specific needs during processing.

Python module

The software has its own Python module that contains five classes:

- FLDataCollection class
- FLDataManager class
- FLDataRecord class
- FLProperties class
- FLPythonString class

Each of these classes has one or more methods.

Supported transforms

You can use the Python module with these transforms:

- Match
- User-Defined

Third-party Python libraries

To ensure that your Python expressions run correctly, make sure that all third-party python libraries are in the appropriate dynamic library path for your operating system so that the dependencies are resolved. If you find that a Python library is not working correctly, update the library path (LD_LIBRARY_PATH for Solaris and Linux, LIBPATH for AIX, and SHLIB_PATH for HP) in the environment where the AL_JobService is installed, and restart the job service.

Processing mode

The User-Defined transform can run in two processing modes. You specify the mode in the User-Defined Editor. The processing mode determines how the Python expression is applied:

- *Per Record:* Applies the expression to each record. This method is useful for formatting your data, such as making the data all uppercase. You cannot add new records into the data flow with this option.

- *Per Collection:* Applies the expression to the entire data collection. For example, the software could go through each record in a collection to decide if a certain statement is true and then take an action on the entire collection. Use this option when adding or deleting new or duplicate records.

Internal coding

For use with Per Record processing mode, much of the Python coding is done internally, so that you don't need to worry about properly importing the Python module. Most of the examples in this section do not include any import syntax.

Unicode in Python

The software is Unicode-enabled. Therefore, all Python methods require Unicode input and all return values are in Unicode. In Python syntax, you must alert Python that you are processing Unicode data, for example (where "u" indicates Unicode):

```
record.SetField(u'NAME', u'value')
```

⚠ Caution

Make sure to use a "u" to indicate Unicode every time you use a Unicode string to look up field names; for example, in a GetField, SetField, or SendToPipe method. If you do not, an error or crash may occur.

Clean up new memory references

Some of the Python methods return new objects when you call the method. You can see which methods return new objects by reviewing the “Return value” heading under each method's section in this document. Any method with a return value returns new objects.

Whenever you create a new object (such as a variable) that refers to one of these methods, you are also creating a new memory reference.

⚠ Caution

Be sure to clean new memory references. If you don't clean up these references in your scripts, you may notice a memory leak after running projects where your script is present.

For example, assume that you want to retrieve the value of a field using the GetField() method, and save that value as a variable. Because that method returns a value, you must clean up that reference by deleting the variable at the end of the script. For example:

```
Master = SRC.GetField(u'input.code')
del Master
```

However, if you use the NewDataRecord() method and do not use Collection.AddRecord(newRec), use the DeleteDataRecord() method before you use the del command. Otherwise, the memory allocation remains. For example:

```
newRec = DataManager.NewDataRecord(1)
DataManager.DeleteDataRecord(newRec)
del newRec
```

If you use Collection.AddRecord(newRec), you do not need to use the DeleteDataRecord() method before you use the del command, because you do not own the record. For example:

```
Collection.AddRecord(newRec)
del newRec
```

The following example gets the first record from a collection and deletes it.

```
newRec = DataManager.NewDataRecord(0)
Collection.GetRecord(newRec, 1)
Collection.DeleteRecord(newRec)
del newRec
```

Using Mapped_Name

When you set up fields for the transforms that use Python scripts in Per Record mode, you can specify the Mapped_Name. The value of this option indicates an “alias” so that you can refer to your field easily in your Python code.

In some transforms, you can specify a Mapped_Name for fields on both input and output. In this case, be sure to use unique values for these options. If you use the same value on both input and output, you will receive an error.

When you use the GetField and SetField methods, make sure the you enter the Mapped_Name correctly in the Python code and map the input field in the transform. If the Mapped_Name is not used correctly and mapped, you may encounter the following error:

FlDataRecord::GetField() error: Invalid field name MAPPED_RECNO.

Using paths with substitution parameters and custom options

Be cautious when writing any sort of expression that might reference a path. The backslash (\) symbol is used in Python to indicate an escaped character. For example, “\u” indicates Unicode. Therefore, if you use certain paths in your expression, it won't be read properly because the escaped character will be recognized as such. For example:

c:*\u*serdata\myfolder

In this specific instance, you would likely receive the following error when you run the project that contains this User-Defined transform: “UnicodeError: Unicode-Escape decoding error: truncated \uXXXX escape”.

It is also important to consider this when using substitution parameters or custom option in your expression. If your substitution parameter refers to a path, it may encounter the same issue.

You can avoid this issue in several different ways:

- Use forward slashes.

```
u"c:/userdata/myfolder/test_file.txt"
```

- Use a double backslash in the path.

```
u"c:\\userdata\\myfolder\\test_file.txt"
```

- Enclose the substitution value with an “r”, which indicates a raw string.

```
r'[$$TEST]'
```

Related Information

Create an expression with the Python Expression editor [page 1817]
Defined classes and methods [page 1821]
FlDataCollection class [page 1822]
FlDataManager class [page 1827]
FlDataRecord class [page 1830]
FlProperties class [page 1832]
FlPythonString class [page 1834]

3.11.1.2 Create an expression with the Python Expression editor

The Python Expression editor, which is similar to the Smart editor, helps you create your Python expressions. The editor provides basic programming features such as keyword highlighting, auto-completion, auto-indentation, and code tool tips.

Related Information

Smart editor [page 1021]
To open the Python Expression editor from the User-Defined transform [page 1818]
Write Python code [page 1818]
Validate syntax [page 1820]
Fix syntax [page 1820]
Find and Replace [page 1820]

3.11.1.2.1 To open the Python Expression editor from the User-Defined transform

Before you use the Python Expression editor, define your input and output fields for the User-Defined transform in the transform editor.

1. In the User-Defined transform editor *Options* tab, click the *Edit Options* button to open the User-Defined Editor.

   You can also select the User-Defined transform in the data flow and choose ▶ *Tools* ▸ *User-Defined Editor* ◀.
2. Decide which processing mode you want to use and select Per Record or Per Collection mode.
3. Select Python Expression Editor in the Option Editor pane, and click the *Launch Python Editor* button.

Related Information

Smart editor [page 1021]

3.11.1.2.2 To open the Python Expression editor from the Match transform

Before you use the Python Expression editor, define your input and output fields for the Match transform in the transform editor.

1. In the Match transform editor *Options* tab, click the *Edit Options* button to open the Match Editor.

   You can also select the Match transform in the data flow and choose ▶ *Tools* ▸ *Match Editor* ◀.
2. Add a best record operation to your Match transform and select the appropriate option values.
3. To customize the Python code, make sure that you select *Yes* in the Custom column of the Best Record Action Fields table. Otherwise, the Python code is not editable.
4. Click the *View/Edit Python* button.

Related Information

Smart editor [page 1021]

3.11.1.2.3 Write Python code

In the Python Expression editor, create and edit Python code in the editor pane of the window. The Python expression that you create here depends on what you need to do with the Match or User-Defined transform.

The Python Expression editor includes keyword highlighting, auto-completion, auto-indentation, and code tool tips. As you type, the Python Expression editor highlights the correct Python syntax. It also auto-completes:

- The Python objects, functions, classes, and methods
- The SAP Data Services generated variables

Python API

The Python API tab lists the objects, functions, classes, and methods that are available for the specific transform and processing mode. When you select an item in the tab, information about it appears in the help area.

Input and output fields

The I/O Fields tab displays the input fields and output fields that have been mapped in the User-Defined transform. You can also add, delete, and edit the properties of user-defined input and output fields from this tab by right-clicking *Input Fields* or *Output Fields* and selecting *Insert*, *Delete*, or *Properties*.

The field variable name is the Mapped_Name option. For example, in a transform you may have an input field as follows:

- *Mapped_Name:* BEST_PRIMNAME1_IN

You could also have an output field as follows:

- *Mapped_Name:* BEST_PRIMNAME1_OUT

The Python Expression editor works in this way for both Per Record and Per Collection processing modes.

Custom options

For the User-Defined transform, you can create custom options that are used as variables within the transform (for example, a file path). You create the custom option in the User-Defined editor, and it is then displayed in the Variables tab of the Python Expression editor. Like substitution parameters, custom options are assigned $$ as a prefix and are enclosed in brackets (for example, [$$PATH]).

If you have both custom options and substitution parameters in your data flow, substitution parameters take precedence over custom options.

Related Information

Smart editor [page 1021]

Python examples [page 1836]

3.11.1.2.4 Validate syntax

When you click the Validate button in the Python Expression editor, the syntax checker makes sure that the Python code has:

- All required colon (:) characters
- All string literal closing characters (either double quotes or single quotes)
- Correct indentation

Validating the Python syntax cannot prevent all runtime errors from occurring. Even if the code is syntactically correct, it might not execute correctly, in which case errors are generated during execution. The syntax checker cannot look for the incorrect usage of:

- Variable names
- Arguments to a function
- Method name on an object

3.11.1.2.5 Fix syntax

If a syntax error is found, a message appears in the bottom section of the edit pane. The message points out the line and character number of the error.

To fix the syntax:

1. Double-click the error message. The Python Expression editor puts the focus on the specified line in the Python code.
2. After you fix an error, click the *Validate* button again. Messages are displayed one at a time; you may have additional syntax errors to fix.
3. Repeat steps 1 and 2 until all of the syntax is correct.

3.11.1.2.6 Find and Replace

Instead of browsing through lines of code, click the *Find* and *Replace* buttons to search for specific text and, if you want, replace it with other text. If the text is found, it is highlighted in the script.

Select the options *Match case* or *Match whole word only* to customize your search.

3.11.1.3 Built-in objects

Data Services includes internally-coded objects that you can use when writing expressions. Make sure to use the exact capitalization as you see it in this documentation. Python is case-sensitive.

The User-Defined transform supports two processing modes:

- Per Record

- Per Collection

Class	Object	Description	User-Defined Record Mode	User-Defined Collection Mode	Best Record (Match transform)
FLData Collection	Collection	Reference to each collection being processed.		Yes	
FlData Record	DST	Reference to destination record in a group posting operation.			Yes
	record	Reference to each record being processed by the data flow.	Yes		
	SRC	Reference to source record in a group posting operation.			Yes
FLData Manager	DataManager	Reference to the data manager for the data flow.		Yes	
FL Properties	Properties	Reference to a properties object.	Yes	Yes	Yes
FLPython String	RET	Value you want to post in a group posting operation.			Yes

3.11.1.4 Defined classes and methods

The following table lists every SAP Data Services-defined class and its supported methods in the User-Defined transform and the Best Record operation of the Match transform. The User-Defined transform supports two processing modes:

- Per Record
- Per Collection

Class	Method	User-Defined Record Mode	User-Defined Collection Mode	Best Record (Match transform)
FlData Collection	AddRecord		Yes	
	DeleteRecord		Yes	
	GetRecord		Yes	
	Size		Yes	

Class	Method	User-Defined Record Mode	User-Defined Collection Mode	Best Record (Match transform)
	Truncate		Yes	
FlData Manager	DeleteDataRecord		Yes	
	NewDataRecord		Yes	
FlData Record	GetField	Yes	Yes	Yes
	SetField	Yes	Yes	Yes
FlProperties	GetProperty	Yes	Yes	Yes
FlPython String	GetBuffer			Yes
	Set Buffer			Yes

3.11.1.5 FlDataCollection class

Use FlDataCollection class methods when you want to manipulate entire collections of data or records, and when adding new records to a collection that did not exist before. This can be helpful when matching in a real-time environment.

Related Information

AddRecord [page 1822]
DeleteRecord [page 1824]
GetRecord [page 1824]
Size [page 1825]
Truncate [page 1826]

3.11.1.5.1 AddRecord

Syntax

```
AddRecord(<record>)
```

Description

Adds the record to the new collection.

Note

For every NewDataRecord(), you can call AddRecord() only once.

Note

After you call AddRecord(), do not call DeleteDataRecord().

Parameters

This method has the following parameter.

Parameter	Description
record	Substitute the name of the record you want to add. This parameter is a variable that you must define.

Return value

None.

Example

In this example, the fields are defined for a new record and then the record is added to the collection with this AddRecord() method.

```
aDup = [1, u'brian', u'boyd', u'123 main st', u'83301'], \
[2, u'bryan', u'boyde', u'456 first st', u'83302'], \
[3, u'brina', u'boyle', u'789 last ave', u'83303']

for rec in aDup:
    newrecord = DataManager.NewDataRecord(1)
    newrecord.SetField(u'ID',unicode(rec[0]))
    newrecord.SetField(u'FIRST_NAME',unicode(rec[1]))
    newrecord.SetField(u'LAST_NAME',unicode(rec[2]))
    newrecord.SetField(u'ADDRESS',unicode(rec[3]))
    newrecord.SetField(u'POSTCODE1',unicode(rec[4]))

    Collection.AddRecord(newrecord)
```

3.11.1.5.2 DeleteRecord

Syntax

```
DeleteRecord(<record>)
```

Description

Removes the specified record from the collection.

Parameters

This method has the following parameter.

Parameter	Description
record	Substitute the name of the record you want to delete. This parameter is a variable that you must define.

Return value

None.

Example

In this example, a new record (newRec) is created and then deleted.

```
newRec = DataManager.NewDataRecord()
Collection.DeleteRecord(newRec)
```

3.11.1.5.3 GetRecord

Syntax

```
GetRecord(<record>, <index>)
```

Description

Retrieves the value of a record in a collection in the specified index position.

Parameters

This method has the following parameters.

Parameter	Description
record	Substitute the name of the record object. This parameter is a variable that you must define; for example, `record = DataManager.NewDataRecord().`
index	Substitute the numerical index value of the record in the collection.

Return value

Returns the value from the record at the specified position.

Example

In the following example, a new record (newRec) is created, the value of the record in position one is retrieved, and the record is deleted from the collection.

```
newRec = DataManager.NewDataRecord()
Collection.GetRecord(newRec, 1)
Collection.DeleteRecord(newRec)
```

3.11.1.5.4 Size

Syntax

```
Size()
```

Description

Counts the number of records in the collection.

Parameters

None.

Return value

Returns an integer that refers to the number of records in the collection.

Example

In this example, you're retrieving the number of records in the collection.

```
collectionSize = Collection.Size()
```

3.11.1.5.5 Truncate

Syntax

```
Truncate()
```

Description

Removes all records from a collection, but does not delete the collection.

Parameters

None.

Return value

None.

Example

Use this method to quickly delete all the records from a collection, rather than one by one.

```
Collection.Truncate()
```

3.11.1.6 FIDataManager class

Use FIDataManager class methods when you want to create new records.

Related Information

DeleteDataRecord [page 1827]
NewDataRecord [page 1828]

3.11.1.6.1 DeleteDataRecord

Syntax

```
DeleteDataRecord(<record>)
```

Description

Deletes the memory of a record object allocated using NewDataRecord().

> Note
>
> Do not call DeleteDataRecord() after calling AddRecord().

Parameters

This method has the following parameter.

Parameter	Description
record	Substitute the name of the record object you want to delete.

Return value

None.

Example

In this example, a new record object (newRec) is created. Then, using this method, the memory allocated to the data collection is deleted. You must use this method when you use the NewDataRecord() method, otherwise the Python expression may have a memory leak.

```
newRec = DataManager.NewDataRecord()
DataManager.DeleteDataRecord(newRec)
```

3.11.1.6.2 NewDataRecord

Syntax

```
NewDataRecord()
```

Description

Creates a new record object. Do not use this method in a loop, otherwise the Python expression may experience a memory leak. Depending on the expression, you'll probably want to place this method at the beginning of the expression.

Parameters

This method has none, but see example for an exception.

If you call the record with a parameter of 1, then the new record gets its own memory.

Return value

Returns a new object of type FIDataRecord.

Examples

In the following example, a new record (newRecord) is created and populated in the original collection.

```
newRecord = DataManager.NewDataRecord()

#gets the number of records
numRecords = Collection.Size()

#iterate over the collection
for recordNum in range (1, numRecords + 1)

  #get a record
  Collection.GetRecord(newRecord, recordNum)

  #set a field on the record
  newRecord.SetField(u'NAME', u'test')

DataManager.DeleteDataManager(newRecord)
```

The following example is a little different from the previous one. In this example, records are read from a database and then are added to the original collection. Because of this difference, the NewDataRecord() method then needs a numeric parameter of 1.

```
newRecord = DataManager.NewDataRecord(1)

#get the records from the database (excluded from example)

#populate the record
newRecord.SetField(u'NAME', u'test')

#add record to the collection
Collection.AddRecord(newRecord)

del newRecord
```

⚠ Caution

Make sure that you clean up memory references.

Related Information

About Python [page 1814]

3.11.1.7 FlDataRecord class

Use Data Services-defined FlDataRecord class methods to manipulate existing individual records.

Related Information

GetField [page 1830]
SetField [page 1831]

3.11.1.7.1 GetField

Syntax

```
fieldVal = GetField(u'<fieldName>')
```

Description

Retrieves the contents of the specified input field. This method can be used with defined input fields only.

Parameters

This method has the following parameter:

Parameter	Description
fieldName	In the Python Expression editor, use one of the input field variables. If you use this method with the Best Record operation of the Match transform, this parameter should be replaced with the Mapped Name you want to retrieve.

Return value

Returns a new string with the contents of the specified field.

Example

```
if newRecord.GetField(u'POSTCODE1') == u'54601'...
```

Caution

Make sure to use a "u" to indicate Unicode every time you use a Unicode string to look up field names. If you do not, an error or crash may occur.

3.11.1.7.2 SetField

Syntax

```
SetField(u'<fieldName>', u'value')
```

Description

Stores a value in the specified field.

Parameters

This method has the following parameters:

Parameter	Description
fieldName	In the Python Expression editor, use one of the input or output field variables, which uses the Mapped Name.
value	Specifies the value you want to store in the field.

Return value

None.

Example

For example, you could store "Current Resident" in the field named NAME_SUBSTITUTION.

```
newRecord.SetField(u'NAME_SUBSTITUTION', u'Current Resident')
```

⚠ Caution

Make sure to use a "u" to indicate Unicode every time you use a Unicode string to look up field names. If you do not, an error or crash may occur.

3.11.1.8 FIProperties class

Use the FIProperties class to gain access to various properties of the system that the Python expression is running in. The class can access the following run-time parameters in the Data Services environment variables:

Run-time parameter	Description
APPLICATION_PATH	The directory that contains the application executable.
APPLICATION_VERSION	The version of the framework.
DATAFLOW_NAME	The name of a data flow.
JOB_ID	The run ID of a job.
TRANSFORM_GUID	The globally unique identifier, or GUID, of a transform
TRANSFORM_NAME	The display name of a transform.
REPOSITORY_VERSION	The version of a repository.

Related Information

GetProperty [page 1833]

3.11.1.8.1 GetProperty

Syntax

```
var1 = GetProperty(<PropertyName>)
```

Description

Returns the value of a given property specified as an input parameter.

Parameters

This method has the following parameter:

Parameter	Description
PropertyName	Specifies the environment variable that you want to retrieve.

Return value

Returns the value of the specified property.

Example

The following example shows how to retrieve a value for the JOB_ID parameter.

```
#Retrieve the Property Value for JOB_ID
propValue = Properties.GetProperty(u'JOB_ID')

#Set the Job Id value into JOB_ID_OUT field
record.SetField(u'JOB_ID_OUT',unicode(propValue))

del propValue
```

3.11.1.9 FlPythonString class

Use the FlPythonString methods to customize your data processing. With these methods, you can create a Best Record operation in the Match transform.

Related Information

GetBuffer [page 1834]
SetBuffer [page 1835]

3.11.1.9.1 GetBuffer

Syntax

```
GetBuffer()
```

Description

Returns the specified string.

Parameters

None.

Return value

The Unicode character string.

Example

In the following example, *getstr* would hold the value of the Unicode character string.

```
getstr = STR.GetBuffer()
```

3.11.1.9.2 SetBuffer

Syntax

```
SetBuffer(u'<stringValue>')
```

Description

Sets character buffer to the object.

Parameters

This method has the following parameter:

Parameter	Description
stringValue	Specifies the string that you want to use here.

Return value

None.

Example

This example shows how to post data from a “master” record to its “subordinates” with the Best Record operation of the Match transform. The data is input with the field input.code.

Best Record strategy:

```
# store master and subordinate values
SOURCE = SRC.GetField(u'input.code')
DESTINATION = DST.GetField(u'input.code')

# if the master is not empty and the subordinate is
if len(SOURCE.strip()) != 0 and len(DESTINATION.strip()) == 0:
   RET.SetBuffer(u'T')
else:
   RET.SetBuffer(u'F')

# delete temporary variables
del SOURCE
del DESTINATION
```

Best Record Action:

```
# store master
SOURCE = SRC.GetField(u'input.code')

# return master
RET.SetBuffer(SOURCE)

# delete temporary variables
del SOURCE
```

3.11.1.10 Python examples

The following examples, grouped by the type of action they perform on data, are intended to help you get started writing expressions in Python. You may need to significantly change some of these examples to fit the type of data and names of fields you are using.

Keep in mind that many of these tasks could be also performed with a Query transform.

Formatting data

The following examples of Python code in the User-Defined transform can be used to format data.

Example use	Sample Python code
Data is input without data-source identification. In the User-Defined transform, append a field mapped to the source and populate it with TRC for all records.	`record.SetField(u'SOURCE',u'TRC')`

Data is input with a name field mapped to name. In the User-Defined transform, upper case the name and put it in a new name field that is mapped to uppername.	`name = record.GetField(u'name')`
	`uppername = name.upper()`
	`record.SetField(u'uppername',unicode(uppername))`
	`del name`
	`del uppername`

Data is input with an account type indicator field, mapped to account_type, that contains B or b for business accounts, and I or I for individual accounts. In the User-Defined transform, append two fields mapped to name and firm. If records contain B or b, output the contents of the field mapped to customer_name to the new firm field. If records do not contain B or b, output the contents to the new name field.	`account_type = record.GetField(u'acct_type')`
	`customer_name = record.GetField(u'cust')`
	
	`if account_type.strip().upper() == u'B':`
	`    record.SetField(u'firm',unicode(customer_name))`
	`else:`
	`    record.SetField(u'name',unicode(customer_name))`
	`del account_type`
	`del customer_name`

Example use	Sample Python code

Firm data is input in a field mapped to firm. In the User-Defined transform, populate a two-character field mapped to firm_length that contains the number of characters in the firm name (padded with zeros).	`field = record.GetField(u'firm')`
	`firm_length = field.strip().zfill(2)`
	`record.SetField(u'firm_length',unicode(firm_length))`
	`del field`
	`del firm_length`

Data is input with some records not having a name in the field mapped to name. In the User-Defined transform, complete empty names with Valued Customer, preserving the input name in records that have them. Overwrite the data in the same field.
In this example, name_in is the mapped name for the input name field. In the output field section, name_out is the mapped name for the same field.
	`name_in = record.GetField(u'name')`
	`if len(name_in.strip()) == 0:`
	`    record.SetField(u'name',u'Valued Customer')`
	`del name_in`

Data is input with fields mapped to ZIP Code and street. In the User-Defined transform, append a field mapped to breakgroupid, and populate it with the first three characters of the ZIP Code and the first three characters of the street.	`zip = record.GetField(u'zip')`
	`street = record.GetField(u'street')`
	`breakgroupid = zip[0:3] + street[0:3]`
	`record.SetField(u'breakgroupid',unicode(breakgroupid))`
	`del zip`
	`del street`
	`del breakgroupid`

Data is input with a field mapped to groupnumber. In the User-Defined transform, append a field mapped to groupnumberzeropad, and populate it with the group number, padded with zeros to 10 characters in length.	`groupnumber = record.GetField(u'groupnumber')`
	`groupnumberzeropad = groupnumber.strip().zfill(10)`
	`record.SetField(u'groupnumberzeropad',unicode(groupnumberzeropad))`
	`del groupnumber`
	`del groupnumberzeropad`

In the User-Defined transform, append a field mapped to recordnum, and populate it with the record number.	`dct = locals()`
	`if dct.has_key('COUNTER'):`
	`    dct['COUNTER'] = dct['COUNTER'] + 1`
	`else:`
	`    dct['COUNTER'] = 1`
	`record.SetField(u'recordnum',unicode(dct['COUNTER']))`

In the User-Defined transform, append a field mapped to recordnum, and populate it with the record number, zero padded to 10 characters in length.	`dct = locals()`
	`if dct.has_key('COUNTER'):`
	`    dct['COUNTER'] = dct['COUNTER'] + 1`
	`else:`
	`    dct['COUNTER'] = 1`
	`recordnum = str(dct['COUNTER']).zfill(10)`

Example use	Sample Python code

	`record.SetField(u'recordnum',unicode(recordnum))`
	`del recordnum`

Data is input in user_group and user_code fields. In the User-Defined transform, if the contents of user_code is A, B, C, D, E, F, G, or H, output User-GroupA in the user_group field. If user_code contains I, J, K, L, M, N, O, or P, output UserGroupB in the user_group field. If user_code contains any other value, preserve the input value in the user_group field.	`user_code = record.GetField(u'user_code')`
	`uga = 'A,B,C,D,E,F,G,H'`
	`ugb = 'I,J,K,L,M,N,O,P'`
	`if uga.find(user_code.strip().upper()) > -1:`
	`    record.SetField(u'user_group',u'UserGroupA')`
	`elif ugb.find(user_code.strip().upper()) > -1:`
	`    record.SetField(u'user_group',u'UserGroupB')`
	`del user_code`
	`del uga`
	`del ugb`

Splitting data

The following examples can be used in the User-Defined transform to split your data in a specific way without changing how it is routed.

Example use	Sample Python code

Data is input in an input account field (mapped to account) with contents of name, a slash, and firm, for example "John Smith / SAP". In the User-Defined transform, append two new fields mapped to name and firm, where the contents before the slash are placed in name and the contents after the slash are placed in firm.
In this example, the syntax str specifies the type of split and the syntax 3-part specifies how to split.
	`from flscansplit import ScanSplit`
	`account = record.GetField(u'account')`
	`name = ScanSplit(account, u'str', u'3-part', ['/'])[0]`
	`firm = ScanSplit(account, u'str', u'3-part', ['/'])[2]`
	`record.SetField(u'name',unicode(name))`
	`record.SetField(u'firm',unicode(firm))`
	`del account`
	`del name`
	`del firm`

Data is input in an input.account field (map-	`from flscansplit import ScanSplit`
	`account = record.GetField(u'account')`

Example use	Sample Python code

ped to account) with contents of a person's name followed by a financial suffix, for example "John Smith JTWROS". In the User-Defined transform, append two new fields mapped to account_name and account_type, where the name and type are split.	`type = [u'JT/WROS', u'JT WROS', u'JTWROS', u'JT/TEN', u'JT TEN', u'JTTEN', u'JT/TIC', u'JT TIC', u'JTTIC', u'TEN COM', u'TEN/COM', u'TENCOM']`
	`account_name = ScanSplit(account, u'str', u'before',type)[0]`
	`account_type = ScanSplit(account, u'str', u'before',type)[1]`
	`record.SetField(u'account_name',unicode(account_name))`
	`record.SetField(u'account_type',unicode(account_type))`
	`del account`
	`del type`
	`del account_name`
	`del account_type`

Another option for scan values is to create an external file in a text editor and saved locally, with extension “py”. Import the file prior to the method. Then in the ScanSplit method, use the variable in the file in place of the actual scan values.

For example, to accomplish the same account_name and account_type fields specified in the second example, you may create a file called suffixes.py that has the following contents:

```
type = [u'JT/WROS', u'JT WROS', u'JTWROS', u'JT/TEN', u'JT TEN', u'JTTEN', u'JT/
TIC', u'JT TIC', u'JTTIC', u'TEN COM', u'TEN/COM', u'TENCOM']
```

Then, complete the following expression in the User-Defined transform.

```
from flscansplit import ScanSplit
from suffixes import *
account = record.GetField(u'account')
account_name = ScanSplit(account, u'str', u'before',type)[0]
account_type = ScanSplit(account, u'str', u'before',type)[1]
record.SetField(u'account_name',unicode(account_name))
record.SetField(u'account_type',unicode(account_type))
del account
del account_name
del account_type
```

Best Record

The following example can be used in the Best Record operation in the Match transform. This example shows the use of Unicode.

Example use	Sample Python code

Data is input with a field gen.phone that is populated in some records of a match group and empty in others. Perform the best record action of taking phone data from a populated record and placing it into an empty record.	*Best Record strategy:*
	`Source = SRC.GetField(u'gen.phone')`
	`Destination = DST.GetField(u'gen.phone')`
	`if len(Source.strip()) > 0 and len(Destina-`

Example use	Sample Python code

In this example, the Best Record strategy returns a True when the source is populated and the destination is empty (or else, it returns a False). At the end, the GetField method places the source data into the destination field, provided the Best Record strategy returns a True.	`tion.strip()) == 0:`
	`    RET.SetBuffer(u'T')`
	`else:`
	`    RET.SetBuffer(u'F')`
	
	`del Source`
	`del Destination`
	
	Best Record Action:
	
	`RET.SetBuffer(SRC.GetField(u'gen.phone')`

Assigning source attributes

The following example can be used in the User-Defined transform to assign a source to records in a collection.

Example use	Sample Python code

Data is input with a field mapped to SOURCE_IN. When the source is CRM or LEADS, assign source attributes. When the source is DoNotMarket, assign list attributes.	`SOURCE_IN = SRC.GetField(u'SOURCE_IN')`
	`if SOURCE_IN.strip() == u'CRM':`
	`  SOURCE_TYPE_OUT = u'N'`
	`  DRIVER_ORDER_OUT = u'020'`
	`  BEST_RECORD_PRIORITY_OUT = u'010'`
	`  INCLUDE_IN_SOURCE_COUNT_OUT = u'Y'`
	`  APPLY_BLANK_PENALTY_OUT = u'Y'`
	`  PERFORM_DATA_SALVAGE_OUT = u'N'`
	`  PROTECT_UNIQUE_ID_OUT = u'Y'`
	`elif SOURCE_IN.strip() == u'Leads':`
	`  SOURCE_TYPE_OUT = u'N'`
	`  DRIVER_ORDER_OUT = u'010'`
	`  BEST_RECORD_PRIORITY_OUT = u'020'`
	`  INCLUDE_IN_SOURCE_COUNT_OUT = u'Y'`
	`  APPLY_BLANK_PENALTY_OUT = u'Y'`
	`  PERFORM_DATA_SALVAGE_OUT = u'N'`
	`  PROTECT_UNIQUE_ID_OUT = u'N'`
	`elif SOURCE_IN.strip() == u'DoNotMarket':`
	`  SOURCE_TYPE_OUT = u'S'`
	`  DRIVER_ORDER_OUT = u'000'`
	`  BEST_RECORD_PRIORITY_OUT = u'000'`
	`  INCLUDE_IN_SOURCE_COUNT_OUT = u'Y'`
	`  APPLY_BLANK_PENALTY_OUT = u'N'`
	`  PERFORM_DATA_SALVAGE_OUT = u'N'`
	`  PROTECT_UNIQUE_ID_OUT = u'N'`
	`record.SetField(u'SOURCE_TYPE_OUT',unicode(SOURCE_TYPE_OUT))`
	`record.SetField(u'DRIVER_ORDER_OUT',unicode(DRIVER_ORDER_OUT))`
	`record.SetField(u'BEST_RECORD_PRIORITY_OUT',unicode(BEST_RECORD_PRIORITY_OUT))`
	`record.SetField(u'INCLUDE_IN_SOURCE_COUNT_OUT',unicode(INCLUDE_IN_SOURCE_COUNT_OUT))`
	`record.SetField(u'APPLY_BLANK_PENALTY_OUT',unicode(APPLY_BLANK_PENALTY_OUT))`
	`record.SetField(u'PERFORM_DATA_SALVAGE_OUT',unicode(PERFORM_DATA_SALVAGE_OUT))`

Example use	Sample Python code

	`record.SetField(u'PROTECT_UNIQUE_ID_OUT',unicode(PROTECT_UNIQUE_ID_OUT))`
	`del SOURCE_TYPE_OUT`
	`del DRIVER_ORDER_OUT`
	`del BEST_RECORD_PRIORITY_OUT`
	`del INCLUDE_IN_SOURCE_COUNT_OUT`
	`del APPLY_BLANK_PENALTY_OUT`
	`del PERFORM_DATA_SALVAGE_OUT`
	`del PROTECT_UNIQUE_ID_OUT`

3.12 Hadoop

Data Services can connect to Apache Hadoop frameworks including HDFS and Hive sources and targets. Data Services must be installed on Linux in order to work with Hadoop. Relevant components of Hadoop include:

- Hadoop distributed file system (HDFS): Stores data on nodes, providing very high aggregate bandwidth across the cluster.
- Hive: A data warehouse infrastructure that allows SQL-like ad-hoc querying of data (in any format) stored in Hadoop.
- Pig: A high-level data-flow language and execution framework for parallel computation that is built on top of Hadoop. Data Services uses Pig scripts to read from and write to HDFS including joins and push-down operations.
- Map/Reduce: A computational paradigm where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster. Data Services uses map/reduce to do text data processing.

3.12.1 Prerequisites

Before configuring Data Services to connect to Hadoop, verify the following prerequisites:

- The machine where the Data Services Job Server is installed is configured to work with Hadoop and has the Pig client installed, and if using Hive, the Hive client. Verify this by logging on to the node and issuing Pig and Hive commands, which should invoke the respective interfaces. For more information, see the "Common commands for correct setup" section.
- The Data Services Job Server does not have to be part of the Hadoop cluster, but Hadoop must be installed on the same machine as the Data Services Job Server and must be configured with access to the cluster machines.
- To ensure that the environment is set up correctly for interaction with Hadoop, the Job Server should start from an environment that has sourced the Hadoop environment script, as follows:

  ```
  source <$LINK_DIR>/hadoop/bin/hadoop_env.sh -e
  ```

- To enable text data processing, ensure the necessary text data processing components have been copied to the HDFS file system, which enables MapReduce functionality. For more information, see “Configuring Hadoop for text data processing”.

Related Information

Administrator Guide: Server Management, To configure Job Servers (Windows) [page 84]
Designer Guide: Transforms, Text Data Processing transforms [page 318]

3.12.1.1 Common commands for correct setup

Here are some useful commands that you can use to check that everything is set up correctly between Hadoop and Data Services. Output shown for various commands may differ, but the important thing to look for is that you don't get errors when you run the command.

Setting up the environment

To set up the Data Services environment for Hadoop, use the following command:

```
$ cd <DS Install Directory>/bin
$ source ./al_env.sh
$ cd ../hadoop/bin
$ source ./hadoop_env.sh -e
```

Checking components

To make sure that Hadoop, Pig, and Hive are set up correctly on the machine where the Data Services Job Server for Hadoop is configured and installed, use the following command:

```
$ hadoop fs -ls /
```

For Hadoop, you should see output similar to the following:

```
$ hadoop fs -ls /
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2013-03-21 11:47 /tmp
drwxr-xr-x   - hadoop supergroup          0 2013-03-14 02:50 /user
```

For Pig, you should see output similar to the following:

```
$ pig
INFO  org.apache.pig.Main - Logging error messages to: /hadoop/pig_1363897065467.log
INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting
to hadoop file system at: hdfs://machine:9000
INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting
to map-reduce job tracker at: machine:9001
grunt> fs -ls /
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2013-03-21 11:47 /tmp
drwxr-xr-x   - hadoop supergroup          0 2013-03-14 02:50 /user
grunt> quit
```

For Hive, you should see output similar to the following:

```
$ hive
Hive history file=/tmp/hadoop/hive_job_log_hadoop_201303211318_504071234.txt
hive> show databases;
OK
default
Time taken: 1.312 seconds
hive> quit;
```

If all commands pass, use `$LINK_DIR/bin/svrcfg` from within the same shell to set up or restart the Job Server.

By running this command you are giving the Job Server the proper environment from which it can start engines that can call Hadoop, Pig, and Hive.

3.12.1.2 Configuring Hadoop for text data processing

Data Services supports text data processing in the Hadoop framework using a MapReduce form of the Entity Extraction transform. To use text data processing in Hadoop, copy the language modules and other dependent libraries to the Hadoop file system (so they can be distributed during the MapReduce job setup) by running the Hadoop environment script as follows:

```
$LINK_DIR/hadoop/bin/hadoop_env.sh -c
```

You only have to do this file-copying operation once after an installation or update, or when you want to use custom dictionaries or rule files. If you are using the Entity Extraction transform with custom dictionaries or rule files, you must copy these files to the Hadoop file system for distribution. To do so, first copy the files into the languages directory of the Data Services installation, then rerun the Hadoop environment script. For example:

```
cp /myhome/myDictionary.nc $LINK_DIR/TextAnalysis/languages
```

```
$LINK_DIR/hadoop/bin/hadoop_env.sh -c
```

Once this environment is set up, in order to have the Entity Extraction transform operations pushed down and handled by the Hadoop system, it must be connected to a single HDFS Unstructured Text source.

Optimizing text data processing for use in the Hadoop framework

When using text data processing in the Hadoop framework, the amount of data a mapper can handle and consequently the number of mappers a job uses, is controlled by the Hadoop configuration setting, `mapred.max.split.size`.

You can set the value for `mapred.max.split.size` in the Hadoop configuration file (located at `$HADOOP_HOME/conf/core-site.xml` or an alternate configuration location, depending on the flavor of Hadoop you are using).

By default, the value for `mapred.max.split.size` is 0, which means that there is no limit and text data processing would run with only one mapper. You should change this configuration value to the amount of data a mapper can handle.

For example, you might have a Hadoop cluster that contains twenty machines and each machine is set up to run a maximum of ten mappers (20 x 10 = 200 mappers available in the cluster). The input data averages 200 GB. If you want the text data processing job to consume 100 percent of the available mappers (200 GB ÷ 200 mappers = 1 GB per mapper), you would set `mapred.max.split.size` to 1073741824 (1 GB).

```
<property>
  <name>mapred.max.split.size</name>
  <value>1073741824</value>
</property>
```

If you want the text data processing job to consume 50 percent of the available mappers (200 GB ÷ 100 mappers = 2 GB per mapper), you would set `mapred.max.split.size` to 2147483648 (2 GB).

Related Information

Entity Extraction transform [page 1502]
HDFS file format [page 933]

3.12.2 Connecting to HDFS

To connect to a Hadoop distributed file system (HDFS), you configure an HDFS file format. You then use the file format as a source or target in a data flow.

Related Information

Reference Guide: HDFS file format [page 933]
Reference Guide: File format [page 920]
Designer Guide: File formats [page 254]

3.12.3 Connecting to Hive

The process to connect to Hive, the data-warehousing infrastructure based on Hadoop, is as follows:

1. Enable the Job Server to support adapters. In the Server Manager, open the configuration editor for the Job Server that is installed on the Hadoop node and select the option *Support adapter and message broker SNMP communication*.
2. In the Administrator, add, configure, and start an adapter instance.
3. In the Designer, add and configure a Hive adapter datastore.

3.12.3.1 Adding, configuring, and starting a Hive adapter instance

Enable the Job Server to support adapters. In the Server Manager, open the configuration editor for the Job Server that is installed on the Hadoop node and select the option *Support adapter and message broker communication*.

Use the Data Services Administrator to add and configure an adapter to connect to Hive.

1. Log in to Data Services Management Console and open the Administrator.
2. Expand the *Adapter Instances* node and click the name of the Job Server that is installed on the Hadoop node.
3. Click the *Adapter Configuration* tab and click *Add.*
4. From the list of installed adapters, click *HiveAdapter*.
5. Enter the following adapter instance properties:
    a) Adapter instance name.
    b) For Classpath, the path to the Hive jar files.

    Click *Apply*.
6. Click the *Adapter Instance Status* tab, select the check box for the adapter, and click *Start.*

The adapter has been configured and now will be available in the Designer for adding a Hive datastore.

In the Designer, add and configure the adapter datastore.

Related Information

Administrator Guide: Server Management, To configure Job Servers on Windows [page 84]
Management Console Guide: Administrator, Adapters [page 1938]
Designer Guide: Datastores, Adapter datastores [page 233]

3.12.3.2 Adding and configuring a Hive adapter datastore

Observe the following prerequisites:

- Enable the Job Server to support adapters. In the Server Manager, open the configuration editor for the Job Server that is installed on the Hadoop node and select the option *Support adapter and message broker communication*.
- Add, configure, and start a Hive adapter instance.

Use the Data Services Designer to add and configure a datastore to connect to Hive as follows.

1. Launch the Designer.
2. In the object library, select the *Datastores* tab.
3. Right-click in the blank area of the object library and click *New*.
4. Configure the datastore as follows:

a) Enter a *Datastore name*.

b) For *Datastore type*, select *Adapter*.

c) Select the Job Server associated with this adapter.

d) Select the adapter instance.

e) Click *Advanced* to display the *Adapter Options*.

f) Enter additional adapter information including the following required options: *Host name* (the name of the machine where the Hive service is running), *Port number* (the port number of the machine where the Hive service is running), and *String size* (defaults to 100).

g) Click *OK*.

You can now browse and/or import metadata from the datastore through the adapter.

Then after building a data flow in a job, configure the Hive source and target object options.

Related Information

Designer Guide: Datastores, Adapter datastores [page 233]
Datastore [page 860]
Hadoop Hive Adapter Source [page 950]
Hadoop Hive Adapter Target [page 974]

3.12.3.3 About partitions

Data Services imports Hive partition columns the same way as regular columns. Partition columns display at the end of the table column list. The column attribute *Partition Column* identifies whether the column is partitioned.

When loading to a Hive target, you can select whether or not to use the *Dynamic partition* option on the *Adapter Target* tab of the target table editor. The partitioned data is evaluated dynamically by Hive when scanning the input data. If *Dynamic partition* is not selected, Data Services uses Hive static loading. All rows are loaded to the same partition. The partitioned data comes from the first row that the loader receives.

Related Information

Hadoop Hive Adapter Target [page 974]

3.13 Reserved Words

3.13.1 About Reserved Words

The following words have special meanings in Data Services and therefore should not be used as names for work flows, data flows, transforms, or other design elements that you create. They should also not be used as user names when you create a Data Services repository. They are reserved with any combination of upper- and lower-case letters.

If you use reserved words you must put double quotation marks around them. For example:

```
"PRIMARY"
```

Reserved words appear in editor text areas in blue.

_AL_DEFINE	_AL_ELSE
_AL_IFDEF	_AL_MESSAGE
_AL_METADATA_ELEMENT	_AL_STORED_PROCEDURE
_AL_TRAN_FUNCTION	_FUNC_TABLE
_MEMORY	_RFC_FUNCTION
_SAP_INNER_JOIN	_SAP_LEFT_OUTER_JOIN
ABAP_PROGRAM	ACTA
ACTAGUICOMMENT	ALGUICOMMENT
ALL	AL_NEST
AL_NESTED_TABLE	AL_PROJECT
AL_REAL_TIME_DATAFLOW	AL_RELATION
AL_REPO_FUNCTION	AL_RFC_SCHEMA_GROUP
AL_UNNEST	AL_UNNEST_SCHEMA_GROUP
AL_UNSPECIFIED_PARAMAND	AND
AS	ASC
BEGIN	BEGIN_SCRIPT
BULK	BY
CALL	CASE

CATCH	CHAR
CHARACTER	CONCAT
CONVERT	CREATE
CUSTOM	
DATABASE	DATAFLOW
DATASTORE	DATE
DATETIME	DECIMAL
DECLARE	DEFAULT
DESC	DISTINCT
DISTINCT_KEY	DOMAIN
DOUBLE	
ELSE	EMBEDDED_DATAFLOW
EMBEDDED_DATAFLOW_RT	END
END_TRY	ERROR
ERROR_CONDITION	ERROR_STEP
FILE	FIRM_NOISE_WORD
FLOWOUTPUT	FOREIGN
FROM	FUNCTION
FUNC_ANY	FUNC_CHAR
FUNC_COL	FUNC_DS
FUNC_NUM	
GENERATED	GLOBAL
GROUP	
HAVING	
IF	IN
INPUT	INT
INTEGER	INTERVAL

IS	
KEY	
LEFTOUTERJOIN	LIKE
LOAD	LONG
LOOKUP	
MOD	
NOT	NULL
NUMERIC	
ON	OR
ORDER	OUT
OUTPUT	
PARALLEL	PIPE
PLAN	PRIMARY
PSFT_TREE	
READ	REAL
REFERENCES	RETURN
RETURNS	ROW
SAP_TREE	SELECT
SESSION	SET
SYSTEM	SYSTEM_PROFILE
TABLE	TIME
TRANSFORM	TRANSFORM_SCHEMA_MAPPING
TRY	
VARCHAR	VARIABLE
VIEW	VOID
WHERE	WHILE

3.14 Glossary

access server	A real-time, request-reply message broker that collects message requests, routes them to a real-time service, and delivers a message reply within the user-specified time.
adapter	An external Data Services interface that is created using the Software Developer Kit or is prebuilt and purchased from SAP.
Administrator	A browser-based application running on the Data Services Management Console that allows you to schedule, execute, monitor batch jobs, and so on.
after-image	The values in an UPDATE row after the row changes, used for log-based changed-data capture (CDC) jobs.
alias	An alternate form or name.
ANKLink	An NCOALink option that provides additional data about moves that occurred in the previous months 19 through 48.
annotation	A note that you can attach to a workspace diagram to describe or explain job, work flow, or data flow.
application	A software program.
association matching	A method that uses the results of two or more match sets to find matches that could not be found within a single match set.
audit point	The object in a data flow where audit statistics are collected.
batch job	A set of objects that you can schedule and execute together.
batch project	A job that executes at a specific time and ends after all the data in the specific source is processed.
before-image	The values in an UPDATE row before the row changes, used for log-based changed-data capture (CDC) jobs.
best record	A record created by consolidating the most complete, accurate, and up-to-date data elements from matching records.
best record priority	A value used to designate data from a particular source as having more or less importance than other data.
Binary Large Object	A field whose data consists of such objects as bitmap graphics, images, OLE objects, and metafiles. See also blob.
blank penalty	A setting that assigns a lower priority to records in which a particular field is blank.
blob	A field whose data consists of such objects as bitmap graphics, images, OLE objects, and metafiles. See also Binary Large Object.
blueprint	A sample job that has already been set up to handle a common business problem.
break group	A subset of records that are more likely to match, and which consist of driver and passenger records. Fields commonly used for creating break groups are postcodes, account or Social Security numbers, or the first two positions (or characters) of the street name.

break key
A user-defined field used to create a break group.

bulk loading
The moving of large amounts of data into a database to achieve optimal performance.

CASS
A USPS certification that requires software vendors to go through a series of tests to prove that their software correctly codes addresses according to USPS requirements, and produces the required USPS reports. See also Coding Accuracy Support System.

CDC
The process of retrieving changes made to a production data source. This process consolidates units of work, ensures, data is synchronized with the original source, and reduces data volume in a warehouse environment. See also changed-data capture.

CDC checkpoint
A software feature that lets you restrict changed-data capture (CDC) subscription reads.

CDC datastore
A datastore that allows you to limit extracted data to changed data only, and connects a changed-data capture (CDC) table in a source databse to Data Services.

CDC subscription
An option on a source CDC table that defines the start and end of your data set, thereby allowing different data flows to extract data from the same table without corrupting data extracted by other data flows.

central repository
A storage mechanism that contains all information normally found in a local repository (definitions for each object in an application), but is optional and is shared by multiple users, who can check objects in and out of the repository.

changed-data capture
The process of retrieving changes made to a production data source. This process consolidates units of work, ensures, data is synchronized with the original source, and reduces data volume in a warehouse environment. See also CDC.

City directory
A file that is used by the USA Regulatory Address Cleanse transform when processing data from the U.S., and contains a table of city names, states, and ZIP Codes, organized by state and city.

classification
An indicator of the types of situations that apply to a word.

client/server
A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

Coding Accuracy Support System
A USPS certification that requires software vendors to go through a series of tests to prove that their software correctly codes addresses according to USPS requirements, and produces the required USPS reports. See also CASS.

Common Warehouse Model
A specification that enables interchange of data warehouse metadata between tools, platforms, and repositories in distributed heterogeneous environments. See also CWM.

compare buffer
A part of memory reserved for processing break groups (one break group at a time) in the Match or Associate transform. A larger buffer typically helps improve performance.

component
A major piece of the software.

conditional
A single-use object, available in work flows, that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

connection string
A string version of the initialization properties needed to connect to a database, also known as a "DSN-less" connection. With a connection string you can easily store connection information or pass it between applications.

content type

The type of data in a field in your data source; helps you map your fields when you set up downstream transforms.

contribution value

A value you assign to a match criteria that represents the importance (or weight) you place on that criteria's data.

custom ABAP program

Software that extracts data from an SAP application using custom logic that is not currently supported by Data Services ABAP generation logic, and generates a data set that you use as a source in a data flow or an ABAP data flow.

custom function

A script you create to evaluate or make calculations on input values and produce a return value.

CWM

A specification that enables interchange of data warehouse metadata between tools, platforms, and repositories in distributed heterogeneous environments. See also Common Warehouse Model.

data collection

A collection of information that is sent between transforms.

data flow

A reusable object containing steps to define the transformation of information from source to target.

data record

A row of data that is constructed at run time.

Data Services engine

The core process that reads job information from the Data Services repository and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The Job Server starts the Data Services engine to execute batch or real-time jobs.

Data Services repository

The database that contains information about a Data Services application. The repository contains information about defined reusable objects, and the metadata for sources and targets, transforms, and functions. The repository also contains the job history and runtime statistics information.

Data Services service

The process that ensures that the Access Server and the Job Server are running.

data set

Rows of data with a defined schema.

data source name

A parameter that provides connectivity for a Windows user to a database through an Open Database Connectivity (ODBC) driver. See also DSN.

data transformation

The phase of the data movement process that occurs between extraction and loading.

data transport

A step in an ABAP data flow that defines a target to store the data set extracted during the flow. You can locate the target file on the SAP Application server or in a location accessible to both the SAP Application server and to Data Services across a network.

data type

The format used to store a value, which can imply a default format for displaying and entering the value.

data validation

The process of defining rules to which correct data should conform. In Data Services you define these rules in the Validation transform.

database link

A communication path from one database server to another.

Database Management System

A software application that builds and maintains database tables. See also DBMS.

DataConnector

An operator instance used to read data files generated by Data Services when performing bulk loading using the Teradata Warehouse Builder.

datastore	A logical channel connecting Data Services to a source or target application.
datastore configuration	The definition of a connection to a particular database from a single datastore.
DBMS	A software application that builds and maintains database tables. See also Database Management System.
debug mode	A state of operation that allows you to diagnose errors while executing a job using the interactive debugging features in the Designer.
degree of parallelism	A property of a data flow that defines how many times each transform defined in the data flow replicates for use on a parallel subset of data. See also DOP.
delimiter	A character sequence used to separate column, row, and text data. To separate columns, a delimiter can be a tab, semicolon, comma, space, or any character sequence. To separate rows of data, a delimiter can be a {new line} or any other character sequence. To denote the start and end of a character string, a delimiter can be single quotation marks ('), double quotation marks ("), or {none}.
Delivery Point Barcode	A form of Postnet barcode, consisting of 62 bars and based on the combination of ZIP Code, ZIP+4, DPBC, and a check digit. See also DPBC.
Delivery Point Validation	A technology that assists you in validating the accuracy of your address information with the USA Regulatory Address Cleanse transform. With DPV you can identify addresses that are undeliverable as addressed and whether an address is a Commercial Mail Receiving Agency (CMRA). See also DPV.
Designer	A graphical user interface that allows you to design and test Data Services jobs.
diacritical character	A character that contains an accent, dieresis (umlaut), tilde, cedilla, or other distinguishing marks (for example, ä or Ç). You can choose to have standardized data with these types of characters. The application uses the Latin-1 code page for assigning these accents.
dictionary	Relational database that contains a lexicon of words and phrases that the data cleansing packages and the Data Cleanse transform use to identify, parse, and standardize data.
directional	A component of the address line that indicates direction, such as North in "211 North 115th St".
disambiguation	The process of resolving ambiguity.
discrete field	Input or output data that has separate fields for each piece of information, such as addresses and names.
discrete line format	Input source format in which pieces of data are parsed down to nearly the most distinct level. For example, a “first name” field would be discrete, whereas a “name” field that could contain first, middle, or last name information would not be discrete.
DOP	A property of a data flow that defines how many times each transform defined in the data flow replicates for use on a parallel subset of data. See also degree of parallelism.
DPBC	A form of Postnet barcode, consisting of 62 bars and based on the combination of ZIP Code, ZIP+4, DPBC, and a check digit. See also Delivery Point Barcode.
DPV	A technology that assists you in validating the accuracy of your address information with the USA Regulatory Address Cleanse transform. With DPV you can identify addresses

that are undeliverable as addressed and whether an address is a Commercial Mail Receiving Agency (CMRA). See also Delivery Point Validation.

drill down
To explore detailed data that was used in creating a summary level of data. How far you drill down depends on the granularity of the data in the warehouse.

driver record
A record that drives the comparison process. Driver records are part of a break group and are compared with passenger records to determine matches.

DSN
A parameter that provides connectivity for a Windows user to a database through an Open Database Connectivity (ODBC) driver. See also data source name.

dual address
A dual address occurs when a record contains two address lines. Two combinations are typical: • PO box and street address: 1000 Main Street, Suite 51 / PO Box 2342 • Rural route or Highway Contract and street address: RR 1 Box 345 / 12784 Old Columbus Road

dual names
Two names included on an address line, such as John and Jane Doe.

Early Warning System
A solution for matching valid delivery points that have been created between updates to the national ZIP+4 directory. EWS uses four months of rolling data found in an intermediate directory that is updated weekly with data from the USPS. See also EWS.

eLOT
A sorting sequence for US mail in which ZIP+4 codes are arranged in the order that they are served by the mail carrier. Compare with Line of Travel (LOT). See also Enhanced Line of Travel.

embedded data flow
A data flow that is called from inside another data flow.

Enhanced Line of Travel
A sorting sequence for US mail in which ZIP+4 codes are arranged in the order that they are served by the mail carrier. Compare with Line of Travel (LOT). See also eLOT.

enterprise application
Software that enables businesses to execute and optimize business and IT strategies in domains like Enterprise Resourcew Planning (ERP), Customer Relationship Management (ERM), or Supply Chain Management (SCM).

enterprise resource planning system
An enterprise application from which Data Services can extract data. See also ERP system.

ERP system
An enterprise application from which Data Services can extract data. See also enterprise resource planning system.

EWS
A solution for matching valid delivery points that have been created between updates to the national ZIP+4 directory. EWS uses four months of rolling data found in an intermediate directory that is updated weekly with data from the USPS. See also Early Warning System.

exception
An error that occurs while executing a job.

expression
A combination of variables, parameters, constants, and functions linked by operation symbols and any required punctuation that describe a rule for calculating a value.

fault code
A numeric value that is assigned to a record after the USA Regulatory Address Cleanse transform validation process that signifies that the particular record was not successfully validated. Each numeric value represent a different type of fault.

file format
A description of how data is or should be organized in a file Data Services reads from or loads to. A file format can be specific to a single file or generic for many files.

Forward Sortation Area	The first three characters of a Canadian alphanumeric postal code, such as K1A in the postal code for Canada Post's Ottawa headquarters, K1A 0B1. See also FSA.
FSA	The first three characters of a Canadian alphanumeric postal code, such as K1A in the postal code for Canada Post's Ottawa headquarters, K1A 0B1. See also Forward Sortation Area.
function	A program that operates on values that are passed to it.
functional area	A virtual group of jobs that relate to the same business function, such as Human Resources or Customers.
gather	To recombine terms, such as alphanumeric terms that you would look up together in the dictionary.
gender	A code that indicates the likelihood of a record being a certain sex. This code is derived from the name and has five possible values: strong male, strong female, weak male, weak female, ambiguous, and unassigned.
generated field	A field that is produced on output by a transform, such as a postcode field generated by the Global Address Cleanse transform.
global suggestion list	A method of completing and populating addresses with minimal data, or offering suggestions for possible matches. This address-entry system is ideal in call center environments or any transactional environment where data cleansing is necessary at the point of entry, and a human being is available to choose one of the suggestions.
host name	A computer's network name (or IP address).
hybrid format	An arrangement for records in which some fields are discrete whereas others are in a multiline format.
impact and lineage analysis	A category of reports on the Management Console that provides end-to-end impact and lineage analysis of Data Services tables and columns and SAP BusinessObjects Enterprise objects such as universes, business views, and reports.
import	The process of acquiring information for the Data Services repository.
input source	The records in a database or file that you want the application to process.
interactive debugger	A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line.
interface	A type of interaction with Data Services which is either internal (allows you to create datastore connections to natively- supported applications), or external (allows Data Services to to communicate with information exchange technologies such as Web Services and MQ queries).
intersource match	A match between records of different sources.
job	The unit of work that can be scheduled independently for execution by the Administrator. Jobs are special work flows that can be scheduled for execution, but cannot be called by other work flows or jobs.
Job Server	The Data Services software that receives requests from the Designer and the Administrator to start and stop jobs.
join rank	A value given to or calculated for all data sets in a data flow by which the application determines which source to read first when assembling the data set in a join.

LACS	A database of addresses that have been permanently converted, usually due to 911 emergency system implementation. The changes often consist of conversion from rural-style addressing to standardized, city-style addressing, or renumbering of existing city-style addresses. See also Locatable Address Conversion System.
lastline	The address information that contains components such as the locality, region, postcode, and sometimes country name.
LDU	The last three characters of a Canadian alphanumeric postal code. Compare with Forward Sortation Area (FSA). See also Local Delivery Unit.
legacy system	An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.
license-controlled feature	A feature that is enabled or disabled depending on the product license.
line	The visual connection between objects in a workspace diagram which shows the left-to-right flow path for data moving between those objects during job execution.
Line of Travel	A sorting sequence for US mail in which ZIP+4 codes are arranged in the order that they are served by the mail carrier. See also LOT.
linked datastores	The datastores in a database link relationship.
Local Delivery Unit	The last three characters of a Canadian alphanumeric postal code. Compare with Forward Sortation Area (FSA). See also LDU.
locale	The identification of a record's world region, which controls the format of data when it is stored, processed, or displayed.
locality	A part of the address line of a record that usually refers to the city or town, but which in some countries, such as the United Kingdom, can extend to include the district.
Locatable Address Conversion System	A database of addresses that have been permanently converted, usually due to 911 emergency system implementation. The changes often consist of conversion from rural-style addressing to standardized, city-style addressing, or renumbering of existing city-style addresses. See also LACS.
lookup table	A table that contains data that other tables can reference with lookup functions that return one or more output columns.
LOT	A sorting sequence for US mail in which ZIP+4 codes are arranged in the order that they are served by the mail carrier. See also Line of Travel.
mail piece unit	A version identifier for printers, representing the unique characteristics of a portion of a mailing.
mapped field	A data-quality-specific field used to tell a data quality transform how to process the data in that field.
master record	The first record in a match group.
match	A pair or group of records that are found to be identical, based on the criteria you set.
match criteria	The options and rules you define for how a match key is used to find records in your data.
match group	A collection of records, consisting of a master and subordinate records, that are found to be matching with each other.

match set	A group of options and rules used to perform comparisons on data.
memory datastore	A datastore connection to an in-memory database.
memory table	An internal table used to store a data set in memory while a job runs.
Meta Integration Model Bridge	A Windows stand-alone utility that converts metadata models among design tool formats. See also MIMB.
metadata	Information acquired and maintained to describe tables in source and target databases.
Metropolitan Statistical Area	An aggregation of counties by the US Office of Management and Budget used for statistical purposes. See also MSA.
MIMB	A Windows stand-alone utility that converts metadata models among design tool formats. See also Meta Integration Model Bridge.
MSA	An aggregation of counties by the US Office of Management and Budget used for statistical purposes. See also Metropolitan Statistical Area.
multiline	A database record format in which address data is not consistently located in the same arrangement in all records.
multiline field	Input or output data that has certain address and name data combined in one field.
NANP	A system for structuring telephone numbers that is shared by 19 North American countries including the United States and Canada. See also North American Numbering Plan.
NCOALink	A USPS product consisting of a secured database of approximately 160 million change of address (COA) records consisting of the names and addresses of individuals, families, and businesses who have filed a change of address with the USPS.
nested data	Information in one table that is related to a single row in another table.
noise word	A word that is meaningless to the matching process.
normal source	An origin of records that the application should consider to be good, eligible records.
North American Numbering Plan	A system for structuring telephone numbers that is shared by 19 North American countries including the United States and Canada. See also NANP.
object	Any item that you create in the Data Services Designer.
object definition	The options that describe the operation of an object, which are viewable in the workspace when you open the object.
object dependent	The state of being associated beneath another object. Any object under the highest level object in the hierarchy is object dependent.
object library	A directory management system that provides access to reusable objects.
object version	An instance of an object. Each time you add or check in an object to the central repository, Data Services creates a new version of the object.
operation code	A flag associated with a row in a data set that indicates the status of the data in the row, such as INSERT, UPDATE, DELETE, and NORMAL.
operational dashboard	A type of report on the Management Console that visually displays the status and performance of job and data flow executions.

Option Editor	A feature in Data Services' Data Quality transform editor through which you can change the value for each option within the transform.
Option Explorer	A pane in the Associate, Match, and User-Defined transform editors which shows a list of the option groups within a transform.
option group	A set of choices that control various business rules for a transform.
other source	In a Match transform, data that should be treated as transparent, such as seed sources, and as such are not counted in determining how to characterize a match group.
parameter	A value passed to a work flow or data flow when that flow is called.
partition	The division of table data into sets based on criteria such as a range or list of values in each row.
passenger record	A row of data in a break group that is compared against the driver record.
pattern file	A plain text file that contains user-defined patterns and is used by the Data Cleanse transform, and can be edited by any text editing program.
PMB	A postal delivery location similar to a post-office box but which is hosted by a private company. See also Private Mail Box.
postal code	A system of letters and/or digits used for sorting mail, such as the numeric ZIP Code used in the U.S. and the alphanumeric FSA LDU system used in Canada.
postcode move	A valid postcode that has been split or moved, so only a portion of the the area that had been covered by the one postcode now has two or more postcodes, including the original one, for the same area.
postcode1	The postal code or five-digit ZIP Code (USA).
postcode2	The secondary part of a postal code, such as the "4051" in the United States postcode "54601-4051".
primary entry	A word or phrase in the dictionary that the data cleansing packages and Data Cleanse transform use to identify, parse, and standardize data.
Private Mail Box	A postal delivery location similar to a post-office box but which is hosted by a private company. See also PMB.
project	The highest-level object in the Designer window, which provides you with a way to organize the other objects you create in Data Services.
projection	An operation within a SELECT statement that the software can push to the database; the subset of columns that you map on the Mapping tag in the query editor.
property	An item of information that describes an object, such as its name, description, or date on which it was created.
query transform	A data transformation object that creates a data set that satisfies conditions you specify.
real-time job	A job that executes on-demand as a "request-response" system.
reference file	A file of address data that Data Services can use to match, assign, standardize, and verify addresses.
relational data	A data set in which data in each column contains a scalar value.

Remote Function Call server
A server that allows third-party programs, including SAP Applications and SAP NetWeaver Business Warehouse, to schedule and initiate Data Services jobs and return the results to Data Services. See also RFC server.

Remote Function Call server interface
The node on the Administrator application of the Data Services Management Console where you configure SAP connections to load data into or read data from an SAP NetWeaver Business Warehouse system. See also RFC server interface.

repository
A set of tables that hold user-created and predefined system objects, source and target metadata, and transformation rules.

request/acknowledge operation
An operation that executes a remote HTTP service in the Request Acknowledge mode, wherein acknowledgement is sent only if the operation is successful.

request/reply operation
An operation that sends a request and then awaits notice of the request's result.

reusable object
An object that can be defined, stored, and reused independent of other objects, and is accessible from the object library.

RFC server
A server that allows third-party programs, including SAP Applications and SAP NetWeaver Business Warehouse, to schedule and initiate Data Services jobs and return the results to Data Services. See also Remote Function Call server.

RFC server interface
The node on the Administrator application of the Data Services Management Console where you configure SAP connections to load data into or read data from an SAP NetWeaver Business Warehouse system. See also Remote Function Call server interface.

rule file
A text file that controls how the application parses data.

rule matching
The process of comparing token classifications against defined rules.

sample size
The number of rows to display in the View Data feature.

sampling rate
The number of rows processed after which Data Services writes information to the monitor log file and updates job events.

sampling rows
The parameter that specifies the frequency with which the Management Console Profiler samples data, beginning with the first row of the specified number of sampling rows.

script
A step in a job or work flow that allows you to calculate values to pass to other parts of the job or work flow by calling functions, executing if-then-else statements, and assigning values to variables.

secondary information
Data that helps the application determine how to process a string in various scenarios.

segment
The format with which the data records of IDocs are interpreted.

server group
An association of Job Servers on different computers that can automatically measure resource availability, and distribute batch jobs or part of a job to the Job Server with the lightest load at run time.

similarity score
A percentage that indicates how much two fields or values are considered alike, which is calculated by the application after the comparison process.

single use object
An object that is defined only within the context of one job or one data flow.

smart editor

A flexible Data Services tool used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

snowbird

An informal term for a person with multiple residences who typically changes where he or she resides according to the season.

source group

A collection of data that you can use to prepare a second set of match statistics, combining the statistics for two or more regular sources.

source record

A row that contains the data you want to use for updating or creating your best record.

standards

Business rules that define how Data Cleanse will apply capitalization or other output formatting to data.

star schema

A database design used to format data in a data mart, and which is based on a single fact table to which any number of dimensional tables may be joined.

step

An object that is part of the definition of a work or data flow, which is represented by an icon in the flow diagram, and is connected to other steps to indicate the flow of data through the data flow, or the order of execution in the work flow.

street address

A postal delivery location that consists of a street name and house number.

subordinate record

Any record in a match group other than the master record.

substitution parameter

A text string "alias" that you can use within your job and transforms, and is defined in a substitution parameter configuration. At runtime, that parameter is replaced with its value anywhere it is used in your job.

substitution parameter configuration

The definition of the substitution parameters used throughout your job in a particular run-time environment.

suggestion list

A group of potential matches presented to the user for selection of the correct one.

suppression source

An origin of data that contains records of information that should be excluded from other output destinations.

table

Database information that is organized into rows and columns that the software reads data from or loads data into.

target

The object into which the application loads extracted and transformed data in a data flow.

TDPID

The server name Data Services uses when loading with the bulk loader option. See also Teradata Director Program ID.

Teradata Director Program ID

The server name Data Services uses when loading with the bulk loader option. See also TDPID.

territory

The locale value for a geographical location (usually the country) where a local language is used.

thread

The instance of the program running on behalf of a process.

tokenization

The creation of tokens, which assigns meaning to each piece of word that results from hyphenation in the Data Cleanse dictionary.

transform	A step in a data flow that acts on a data set, and is available through the object library in three cateogories: Data Integrator, Data Quality, and Platform.
try/catch block	A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a job.
unique identifier	In a Data Quality transform, an ID that is unique to a record or group of matching records, and is sequential, static, and will not change when records are updated or re-processed through the application.
unique record	A record that does not have any matching or subordinate records and so does not belong to any match group after the matching process is complete.
web service request	Any message sent from a web client that requires processing by a real-time job.
web services	A platform on which multiple applications can communicate with each other even though constructed in different languages and on different platforms.
weighted scoring	A method of comparison that lets you use values to place more or less importance on various match criteria during the matching process.
work flow	A reusable object that contains steps defining the order of job execution.
workspace	The area on the Designer window where you can manipulate system objects and graphically assemble data movement processes.
Z4Change	A directory of all U.S. ZIP Codes and ZIP+4 Codes, in which those codes that have changed in the last 12 months are flagged.
ZCF	A directory that is used by the USA Regulatory Address Cleanse transform when processing data from the U.S., and contains a table of city names, states, and ZIP Codes, organized by ZIP Code. See also ZIP City File.
ZIP City File	A directory that is used by the USA Regulatory Address Cleanse transform when processing data from the U.S., and contains a table of city names, states, and ZIP Codes, organized by ZIP Code. See also ZCF.
ZIP plus 4	A postal code that consists of both the USPS's 5-digit ZIP Code and the 4-digit add-on code.

4 Management Console Guide

4.1 Introduction

4.1.1 Welcome to SAP Data Services

4.1.1.1 Welcome

SAP Data Services delivers a single enterprise-class solution for data integration, data quality, data profiling, and text data processing that allows you to integrate, transform, improve, and deliver trusted data to critical business processes. It provides one development UI, metadata repository, data connectivity layer, run-time environment, and management console—enabling IT organizations to lower total cost of ownership and accelerate time to value. With SAP Data Services, IT organizations can maximize operational efficiency with a single solution to improve data quality and gain access to heterogeneous sources and applications.

4.1.1.2 Documentation set for SAP Data Services

You should become familiar with all the pieces of documentation that relate to your SAP Data Services product. The latest Data Services documentation can be found on the *SAP Help Portal*.

Document	What this document provides
Adapter SDK Guide	Information about installing, configuring, and running the Data Services Adapter SDK
Administrator Guide	Information about administrative tasks such as monitoring, lifecycle management, security, and so on.
Customer Issues Fixed	Information about customer issues fixed in this release. Note In some releases, this information is displayed the Release Notes.
Designer Guide	Information about how to use Data Services Designer.
Documentation Map	Information about available Data Services books, languages, and locations.
Installation Guide for Windows	Information about and procedures for installing Data Services in a Windows environment.
Installation Guide for UNIX	Information about and procedures for installing Data Services in a UNIX environment.
Integrator Guide	Information for third-party developers to access Data Services functionality using web services and APIs.

Document	What this document provides
Master Guide	Information about the application, its components and scenarios for planning and designing your system landscape. Information about SAP Information Steward is also provided in this guide.
Management Console Guide	Information about how to use Data Services Administrator and Data Services Metadata Reports.
Performance Optimization Guide	Information about how to improve the performance of Data Services.
Reference Guide	Detailed reference material for Data Services Designer.
Release Notes	Important information you need before installing and deploying this version of Data Services.
Technical Manuals	A compiled, searchable, "master" PDF of core Data Services books: ● *Administrator Guide* ● *Designer Guide* ● *Reference Guide* ● *Management Console Guide* ● *Performance Optimization Guide* ● *Integrator Guide* ● *Supplement for J.D. Edwards* ● *Supplement for Oracle Applications* ● *Supplement for PeopleSoft* ● *Supplement for Salesforce.com* ● *Supplement for Siebel* ● *Supplement for SAP* ● *Workbench Guide*
Text Data Processing Extraction Customization Guide	Information about building dictionaries and extraction rules to create your own extraction patterns to use with Text Data Processing transforms.
Text Data Processing Language Reference Guide	Information about the linguistic analysis and extraction processing features that the Text Data Processing component provides, as well as a reference section for each language supported.
Tutorial	A step-by-step introduction to using Data Services.
Upgrade Guide	Information to help you upgrade from previous releases of Data Services and release-specific product behavior changes from earlier versions of Data Services to the latest release.
What's New	Highlights of new key features in this SAP Data Services release. This document is not updated for support package or patch releases.
Workbench Guide	Provides users with information about how to use the Workbench to migrate data and database schema information between different database systems.

In addition, you may need to refer to several Supplemental Guides.

Document	What this document provides
Supplement for SAP	Information about interfaces between Data Services, SAP Applications, SAP Master Data Services, SAP NetWeaver BW, and SAP Master Data Services.
Supplement for SuccessFactors	Information about interfaces between Data Services and SuccessFactors.
Supplement for Salesforce.com	Information about how to install, configure, and use the SAP Data Services Salesforce.com Adapter Interface.
Supplement for J.D. Edwards	Information about interfaces between Data Services and J.D. Edwards World and J.D. Edwards OneWorld.
Supplement for Oracle Applications	Information about the interface between Data Services and Oracle Applications.
Supplement for PeopleSoft	Information about interfaces between Data Servicesand PeopleSoft.
Supplement for Siebel	Information about the interface between Data Services and Siebel.

We also include these manuals for information about SAP BusinessObjects Information platform services.

Document	What this document provides
Information platform services Administrator Guide	Information for administrators who are responsible for configuring, managing, and maintaining an Information platform services installation.
Information platform services Installation Guide for UNIX	Installation procedures for SAP BusinessObjects Information platform services on a UNIX environment.
Information platform services Installation Guide for Windows	Installation procedures for SAP BusinessObjects Information platform services on a Windows environment.

4.1.1.3 Accessing documentation

You can access the complete documentation set for SAP Data Services in several places.

4.1.1.3.1 Accessing documentation on Windows

After you install SAP Data Services, you can access the documentation from the Start menu.

1. Choose *Start* > *Programs* > *SAP Data Services 4.2* > *Data Services Documentation* > *All Guides* .
2. Click the appropriate shortcut for the document that you want to view.

4.1.1.3.2 Accessing documentation on UNIX

After you install SAP Data Services, you can access the documentation by going to the directory where the printable PDF files were installed.

1. Go to `<LINK_DIR>/doc/book/en/`.
2. Using Adobe Reader, open the PDF file of the document that you want to view.

4.1.1.3.3 Accessing documentation from the Web

You can access the complete documentation set for SAP Data Services from the SAP Business Users Support site.

To do this, go to *http://help.sap.com/bods*.

You can view the PDFs online or save them to your computer.

4.1.1.4 SAP information resources

A global network of SAP technology experts provides customer support, education, and consulting to ensure maximum information management benefit to your business.

Useful addresses at a glance:

Address	Content
Customer Support, Consulting, and Education services *http://service.sap.com/*	Information about SAP support programs, as well as links to technical articles, downloads, and online forums. Consulting services can provide you with information about how SAP can help maximize your information management investment. Education services can provide information about training options and modules. From traditional classroom learning to targeted e-learning seminars, SAP can offer a training package to suit your learning needs and preferred learning style.
Product documentation *http://help.sap.com/bods/*	SAP product documentation.
Supported Platforms (Product Availability Matrix) *https://service.sap.com/PAM*	Get information about supported platforms for SAP Data Services. Use the search function to search for Data Services. Click the link for the version of Data Services you are searching for.
SAP Data Services Community Network *http://scn.sap.com/community/data-services*	Get online and timely information about SAP Data Services, including forums, tips and tricks, additional downloads, samples, and much more. All content is to and from the community, so feel free to join in and contact us if you have a submission.
Blueprints *http://scn.sap.com/docs/DOC-8820*	Blueprints for you to download and modify to fit your needs. Each blueprint contains the necessary SAP Data Services project, jobs, data flows, file formats, sample data, template

Address	Content
	tables, and custom functions to run the data flows in your environment with only a few modifications.
SAPTerm *https://portal.wdf.sap.corp/go/sapterm*	SAP's terminology database, the central repository for defining and standardizing the use of specialist terms.

4.1.2 Overview of this guide

4.1.2.1 About this guide

The guide includes information about the SAP Data Services Management Console, a collection of Web-based applications for administering Data Services jobs and services, viewing object relationships, evaluating job execution performance and data validity, and generating data quality reports.

You can install the Management Console on a separate computer from other Data Services components. It runs on your web application server. The Management Console is written in Java and uses a JDBC connection to repositories.

The Management Console includes the following applications:

- *Administrator*
  Use to manage your production environment including batch job execution, real-time services, Web services, adapter instances, server groups, central and profiler repositories, and more.
- *Impact and Lineage Analysis*
  Use to analyze the end-to-end impact and lineage for Data Services tables and columns and SAP BusinessObjects Business Intelligence platform objects such as universes, business views, and reports.
- *Operational Dashboard*
  Use to view dashboards of job execution statistics to see at a glance the status and performance of your job executions for one or more repositories over a given time period.
- *Data Validation Dashboard*
  Use to evaluate the reliability of your target data based on the validation rules you created in your batch jobs to quickly review, assess, and identify potential inconsistencies or errors in source data.
- *Auto Documentation*
  Use to view, analyze, and print graphical representations of all objects as depicted in the Designer including their relationships, properties, and more.
- *Data Quality Reports*
  Use to view and export reports for batch and real-time jobs such as job summaries and data quality transform-specific reports.

4.1.2.2 Who should read this guide

This and other SAP Data Services documentation assume the following:

- You are an application developer, consultant or database administrator working on data extraction, data warehousing, data integration, or data quality.
- You understand your source and target data systems, DBMS, legacy systems, business intelligence, and messaging concepts.
- You understand your organization's data needs.
- You are familiar with SQL (Structured Query Language).
- If you are interested in using this software to design real-time processing, you are familiar with:
    - DTD and XML Schema formats for XML files
    - Publishing Web Services (WSDL, REST, HTTP/S and SOAP protocols, etc.)
- You are familiar with SAP Data Services installation environments: Microsoft Windows or UNIX.

4.2 Logging into the Management Console

When you log in to the Management Console, you must log in as a user defined in the Central Management Server (CMS).

1. Navigate to the Management Console address in a supported web browser:

   ```
   http://<hostname>:28080/DataServices
   ```

   where **<hostname>** is the name of the computer hosting the web application server.

   > i Note
   >
   > If you are logged in to the Designer, you can also access the Management Console home page in several ways:
   >
   > - From the Start page, click *Data Services Management Console*.
   > - From the *Tools* menu, click *Data Services Management Console*.
   > - Click the *Data Services Management Console* tool bar icon.

2. Enter your user credentials for the CMS.
    - *System*
      Specify the server name and optionally the port for the CMS.
    - *User name*
      Specify the user name to use to log into CMS.
    - *Password*
      Specify the password to use to log into the CMS.
    - *Authentication*
      Specify the authentication type used by the CMS.
3. Click *Log on*.
   The software attempts to connect to the CMS using the specified information. When you log in successfully, the list of local repositories that are available to you is displayed.

The Management Console home page opens.

4.2.1 Management Console navigation

After logging in to the Management Console and launching one of the applications, the application name appears under the Management Console banner.

The upper-right side of the main window includes the following links:

- *Home*
  Click to return to the Management Console home page (for example, to select another application).
- *Settings*
  The metadata reporting applications also include a Settings control panel for changing a variety of options depending on the selected application.
- *Logout*
  Click to exit the application and the Management Console and return to the login page.
- Help icon
  Opens the Management Console Guide.

As you navigate around the applications, notice that the top of the right pane often displays a “bread crumb” path to indicate where you are in the application. Depending on the page displayed, sometimes you can click on the bread crumbs to navigate to a different part of the application.

The Administrator, Impact and Lineage Analysis, and Auto Documentation applications also use a navigation tree in the left pane.

Management Console sessions time out after 120 minutes (2 hours) of inactivity.

4.3 Administrator

This section describes the Administrator and how to navigate through its browser-based, graphical user interface.

Use the Administrator to:

- Set up users and their roles
- Add connections to Access Servers and repositories
- Access job data published for Web Services
- Schedule and monitor batch jobs
- Configure and monitor:
  - Access Server status
  - Real-time services
  - Client interfaces including SAP application client interfaces (to read IDocs) and message traffic moving in and out of an Access Server
  - Adapter instances (a prerequisite for creating adapter datastores)

Related Information

Logging into the Management Console [page 1867]

Administrator navigation [page 1869]

4.3.1 Administrator navigation

The layout of the Administrator consists of a window with a navigation tree on the left and pages with tabs on the right.

4.3.1.1 Navigation tree

The navigation tree is divided into several nodes, including:

- Status
- Batch
- Real-Time
- Web Services
- SAP Connections
- Server Groups
- Profiler Repositories
- Management

> Note
>
> The nodes displayed depend on the permissions granted to the user you use to log in to the Management Console. For example, the Profiler Repositories node only appears if you have access to view or manage a profiler repository, as defined in the Central Management Console (CMC).

4.3.1.1.1 Status node

When the Administrator opens, it displays the Status page. The Status page displays the status of the following items (after you have connected them to the Administrator). The red, green, and yellow icons indicate the overall status of each item based on the jobs, services, and other objects they support.

- *Batch*
  Contains the name of the repository associated with the Job Server on which you run the batch jobs. To see batch jobs status, connect the repository to the Administrator.
  Click the repository name to display a list of batch jobs and their status.
- *Real-Time*
  Contains the name of the Access Servers associated with a real-time service. To see real-time jobs status, connect the Access Server to the Administrator.
  Click the Access Server name to display a list of real-time services and their client interfaces.
- *Adapters*

Contains the name of the repository associated with the Job Server on which you run the adapter. To see an adapter's status, enable a Job Server for adapters, and then add the repository associated with that Job Server.

- *Profiler*
  Contains the name of the repository associated with the Profiler Server. To see a profiler repository, connect the profiling repository to the Administrator.
  Click the repository name to display a list of profiler tasks and their status.

4.3.1.1.2 Batch node

After you add at least one repository connection to the Administrator, you can expand the *Batch* node. Then click a repository name to display its Batch Job Status page.

Click the *All Repositories* option to see jobs in all repositories connected to this Administrator (this node only appears if more than one repository is connected).

Each repository under the Batch node includes the following tabs:

- *Batch Job Status*
  View the status of the last execution and in-depth information about each job.
- *Batch Job Configuration*
  Configure execution and scheduling options for individual jobs.
- *Repository Schedules*
  View and configure schedules for all jobs in the repository.

Related Information

Batch Jobs [page 1890]

4.3.1.1.3 Real-Time node

After you add a connection to an Access Server in the Administrator, you can expand the *Real-Time* node. Expand an Access Server name under the Real-Time node to view the options.

Access Server node options	Description
Status	View status of real-time services and client interfaces supported by this Access Server. Control, restart, and set a service provider interval for this Access Server.
Real-time Services	View status for services and service providers, start and stop services, add or remove a service, configure Job Servers for a service.
Client Interfaces	View status for client interfaces, start and stop interfaces, add or remove an interface.

Access Server node options	Description
Logs - Current	View list of current Access Server logs, content of each log, clear logs, configure content of logs for display, enable or disable tracing for each Access Server.
Logs - History	View list of historical Access Server logs, view content of each log, delete logs.

Related Information

Real-Time Jobs [page 1905]
Real-Time Performance [page 1920]

4.3.1.1.4 Web Services node

Use this node to select real-time and batch jobs that you want to publish as Web service operations and to monitor the status of those operations. You can also use the node to set security for jobs published as Web service operations and view the WSDL file that SAP Data Services generates.

Related Information

Support for Web Services [page 1945]

4.3.1.1.5 Adapter Instances node

Use this node to configure a connection between SAP Data Services and an external application by creating an adapter instance and dependent operations. This is a prerequisite requirement for creating a datastore for adapters in the Designer.

After you create a datastore, import data through the adapter and create jobs. Then use this node to view the status of Adapter instances. Options are listed by Job Server under the Adapter Instance node.

Related Information

Adapters [page 1938]

4.3.1.1.6 Server Groups node

The Server Groups node allows you to group Job Servers that are associated with the same repository into a server group.

Use a server group if you want SAP Data Services to automatically use the Job Server on a computer with the lightest load when a batch job is executed. This functionality improves load balancing (throughput) in production environments and also provides a hot backup method. When a job is launched, if a Job Server is down, another Job Server in the same group executes the job.

Related Information

Server Groups [page 1885]

4.3.1.1.7 Profiler Repositories node

After you connect a profiler repository to the Administrator, you can expand the *Profiler Repositories* node. Click a repository name to open the Profiler Tasks Status page.

Related Information

Profile Server Management [page 1929]

4.3.1.1.8 Management node

The Management node contains the configuration options for the Administrator application. Before you can use some features of the Administrator, you must add connections to other SAP Data Services components using the Management node. For example, expand the management node and:

- Expand *Datastore* and click a repository to manage datastore configurations for that repository.
- Click *Access Servers* to add a connection to your Access Servers (for real-time jobs).

Related Information

Administrator Management [page 1873]

4.3.1.2 Pages

The top of the page indicates the currently selected node. Once you select a branch on the navigation tree to go to a page, use the tab row on the page to navigate further.

As you drill into various pages, a “bread crumb” trail often indicates where you are in the Administrator application. Depending on the page displayed, sometimes you can click on the bread crumb links to navigate to a different page.

A dark blue (shaded) tab signifies the active page. Click a light blue tab to go to that page. Some pages do not include a tab row.

4.3.2 Administrator Management

Use the Management features to configure the Administrator.

4.3.2.1 Managing database account changes

SAP Data Services uses several types of user accounts and associated passwords. For various reasons, database account parameters such as user names or passwords change. For example, perhaps your company's compliance and regulations policies require periodically changing account passwords for security.

4.3.2.1.1 Updating local repository login parameters

If the login information, particularly the password, for a repository has changed, SAP Data Services provides an optional password file that all schedules or exported execution commands use. In other words, the software uses this password file to store and update connection information in one location that multiple schedules or exported execution commands share for that repository.

> Note
>
> This description does not apply to central repositories.

The password file:

- Specifies the connection information for the repository.
- Can be stored in a central location for access by others who run jobs in that repository.
- Is created when you create or update a job schedule to minimize associated maintenance.

Related Information

Using a third-party scheduler [page 1899]

4.3.2.1.1.1 Updating the CMS connection information and use a password file

1. Expand the *Management* node.
2. Click *CMS Connection*.
3. Edit the connection information as necessary.
4. Click *Apply*.
5. Click *Generate password file* to create or update the password file.

   The default name and location of the file are `<DS_COMMON_DIR>\conf\repositoryname.txt`.

4.3.2.1.1.2 Updating job schedules

When database account information for your repository changes, the SAP Data Services job schedules associated with that account must also be updated. When you use a password file, the job schedules access it at runtime to automatically retrieve the updated account information.

Related Information

Scheduling jobs [page 1891]

4.3.2.1.2 Updating datastore connection parameters

If the information associated with a datastore connection changes, particularly passwords, you can update the changes using the Administrator.

> i Note
>
> Only users with Administrator role privileges can edit datastore parameters.

4.3.2.1.2.1 Editing the connection information for an individual configuration in a datastore

1. Select *Management* > *Datastore* , and select the repository that contains the datastore configuration that you want to edit.
2. Click the configuration name to configure.
3. Edit the enabled fields as necessary.

4. Click *Apply*. To return all fields to the last set of values applied, click *Reset*.

4.3.2.1.2.2 Editing the connection information for multiple configurations in a datastore

1. Select *Management* > *Datastore* , and select the repository that contains the datastore configurations that you want to edit.
2. Click the datastore name to configure.

   All configurations for that datastore display.
3. Edit the enabled fields as necessary.

   Click *More* to display the page for that individual configuration, which includes more options specific to it.
4. Click *Apply*. To return all fields to the last set of values applied, click *Reset*.

4.3.2.2 Configuring the report server

For each repository registered in the Central Management Console (CMC), a report server configuration is automatically created with default parameters. The Report Server Configuration node in the Management Console Administrator lets you edit the default parameters, including the location where job reports are written upon execution.

1. Select *Management* > *Report Server Configuration* > **`<repository>`** .
   The Report Server Configuration page opens and displays the report export configuration parameters. If the configuration has not yet been saved for the selected repository, the page displays default parameters.
2. Enter the appropriate configuration information.

Option	Description
Host name	The name of the machine that the report server is running on. By default, the current web application server name is used. Localhost is not a valid name.
Communication port	The port number of the machine that the report server is running on.
Export location	The path where the reports will be exported to. The default path is **`<DS_COMMON_DIR>`**`\DataQuality\reports\`. Upon execution, the repository name and job name folders are appended to the path. If the Overwrite option is not selected, a run ID folder is also appended to the path. i Note If you export reports to a location other than a local drive, such as a network drive, before you execute the job you must start the web application server with an account that has access rights to that location.
Export type	The format in which the reports can be exported (PDF or RTF).

Option	Description
Overwrite	Specifies whether existing reports will be overwritten when the reports are exported. If this option is not selected, the reports are exported to a subfolder with the run ID, which specifies a unique identification of an instance of the executed job.
Language	The supported language that the reports are generated in. Note that some reports, such as country-specific certification reports, are designed only to support English, so changing the option for those reports has no effect.

3. Click *Apply* to save the configuration. To return all fields to the last set of values applied, clicking *Reset*.
4. Verify that the security setting for this operation is disabled. Select *Administrator* > *Web Services* and click the *Web Services Configuration* tab. If the Export_DQReport operation is enabled (displays a check in the Session Security column), select the checkbox next to it, select *Disable Session Security* from the pull-down menu, and click the *Apply* button.

To generate and export all of the job reports to the specified location at runtime, select the Export Data Quality Reports option when you execute the job.

Related Information

Adding a job schedule [page 1892]

Reference Guide: Batch Job, Execution options [page 840]

Integrator Guide: To configure web service information using the Administrator [page 2221]

4.3.2.3 Adding Access Servers

The Administrator acts as a front end for Access Servers connected to it. Use the Administrator to:

- Configure real-time jobs as real-time services.
- Configure real-time services with service providers.
- Monitor Access Servers, real-time services, and service providers.

You first must connect an Access Server to the Administrator so that you can use the Administrator to create a real-time service from a real-time job. After a service starts, the Access Server brokers messages between external applications and SAP Data Services.

When a message request comes in, the Access Server communicates with the Job Server to get the repository data needed to run a real-time service and process the message. A reply comes back through the Access Server to the message originator and the Access Server log records the event, which you can monitor from the Administrator.

Use the Access Servers page to connect an Administrator to a repository.

1. Select *Management* > *Access Servers*.
2. Click *Add*.
3. Enter the following information.

Option	Description
Machine Name	Host name of the computer on which the Access Server is installed.
Communication Port	Port assigned to the Access Server in the Server Manager utility.

4. (Optional) Before attempting to register the Access Server with the Administrator, click *Ping* to see if the Access Server is available and exists on the computer and port you specified.
5. Click *Apply*.

   The Administrator registers the Access Server, validates the Access Server connection information, and displays the information on the Access Servers page.

To view a list of Access Servers connected to the Administrator, select *Management* > *Access Servers*.

The Access Servers page lists the Access Servers that are connected to the Administrator. You can also remove a connection to an Access Server from this page.

4.3.2.4 Setting the status interval

Use the Status Interval page to specify the time period for which the Administrator displays the status (using the red, yellow, and green status icons) on the Batch Job Status page.

1. Select *Management* > *Status Interval*.
2. On the Status Interval page, specify the time period.

   You can filter the information on this page in three ways:

   ○ By the last execution of each job
   ○ By number of days
   ○ By range of dates
3. Click *Apply*.

   The Administrator updates the list of job executions and the status interval displays in the table title on the Batch Job Status page. The following example lists the last execution of all jobs.

4.3.2.5 Exporting certification logs

When you run address cleanse jobs with the appropriate options set, you can generate reports to qualify for mailing discounts with certain countries' postal authorities. In the Management Console, you can generate the certification log files required for those certifications.

The Certification Logs page is available to users who are assigned either the Administrator or Operator role.

Related Information

Exporting NCOALink certification logs [page 1878]

Exporting New Zealand SOA certification logs [page 1879]
Exporting DSF2 certification log [page 1879]

4.3.2.5.1 Exporting NCOALink certification logs

Before you export the certification log, you must have run a job containing a USA Regulatory Address Cleanse transform with the NCOA certification options set appropriately. You must also configure your repository in the Central Management Console (CMC).

You can export the certification log for the data in one repository or in all repositories.

> ⚠ Caution
>
> If you select all repositories and have more than one connection to the same repository, your results may contain duplicate records.

1. Select *Management* ▸ *Certification Logs*, and select the repository that contains the certification log that you want to export.
2. Click the *NCOALink* tab.
3. Select the date range that contains the data that you want to export.
4. Select the NCOALink licensee ID for the logs that you want to export, or select All NCOA Licensee IDs.
5. Specify the location where the certification logs will be exported.

   > ℹ Note
   >
   > The location that you specify is relative to the web application server.

   To reset the export location to the default, click the *Reset* button. The default location is `<DS_COMMON_DIR>\DataQuality\certifications\CertificationLogs\<repository>\`.
6. If you want to overwrite an existing log file, click the *Overwrite* option.
7. Click the *Search* button. The page displays the available log files with the specified criteria. You can sort the log files using the column headers.

   The *Data Available* column has a Yes status when there is data for at least one of the log types (PAF, Bala, or CSL). A No status indicates that no data is available and will result in an empty log file for each type.
8. Select the log file(s) that you want to export or select the checkbox next to *Select All*.
9. Click the *Export* button.
   After the log file is exported, a confirmation message is displayed at the top of the page.

Related Information

Administrator Guide: Repository management, To register a repository in the CMC [page 52]
NCOALink Processing Summary Report [page 2000]
Designer Guide: Data Quality, Address Cleanse, NCOALink (USA Regulatory Address Cleanse) [page 650]
Reference Guide: Transforms, NCOALink options [page 1326]

4.3.2.5.2 Exporting New Zealand SOA certification logs

Before you export the certification log, you must have run a job containing a Global Address Cleanse transform with the New Zealand SOA certification options set appropriately. You must also configure your repository in the CMC.

You can export the certification log for the data in one repository or in all repositories.

> ⚠ Caution
>
> If you select all repositories and have more than one connection to the same repository, your results may contain duplicate records.

1. Select ▶ *Management* ▶ *Certification Logs* ◀, and select the repository that contains the certification log that you want to export.
2. Click the *New Zealand SOA* tab.
3. Select whether you want to export all certification log data that is in the selected repository or just the data within a specified date range.

   The Year list contains the current year and the two previous years, because certification logs are required to be retained for two years.
4. Specify the location where the certification logs will be exported.

   The default location is **<DS_COMMON_DIR>**`\DataQuality\certifications\CertificationLogs\`**<repository>**`\`. To reset the export location to the default, click the *Reset* button.
5. If you want to overwrite an existing log file, click the *Overwrite* option.
6. Click the *Export* button.
   After the log file is exported, a confirmation message is displayed at the top of the page.

Related Information

New Zealand Statement of Accuracy (SOA) report [page 2012]
Designer Guide: Data Quality, Address Cleanse, New Zealand Certification [page 613]
Reference Guide: Transforms, Global Address engine options [page 1214]
Reference Guide: Transforms, Report options for New Zealand [page 1217]

4.3.2.5.3 Exporting DSF2 certification log

Before you export the certification log, you must have run a job containing a USA Regulatory Address Cleanse transform with the DSF2 certification options set appropriately. You must also configure your repository in the Central Management Console (CMC).

You can export the certification log for the data in one repository or in all repositories.

⚠ Caution

If you select all repositories and have more than one connection to the same repository, your results may contain duplicate records.

1. Open the Data Services Management Console.
2. Click the *Administrator* icon.
3. Select ▶ *Management* ▶ *Certification Logs* ◀, and select the repository that contains the certification log that you want to export.
4. Click the *DSF2* tab.
5. Select the date range that contains the data that you want to export.
6. Select the DSF2 licensee ID drop list and select the logs that you want to export, or select All DSF2 Licensee IDs.
7. Specify the location where the certification logs will be exported.

   i Note

   The location that you specify is relative to the web application server.

   To reset the export location to the default, click the *Reset* button. The default location is **`<DS_COMMON_DIR>`**`\DataQuality\certifications\CertificationLogs\`**`<repository>`**`\`.
8. If you want to overwrite an existing log file, click the *Overwrite* option.
9. Click the *Search* button. The page displays the available log files with the specified criteria. You can sort the log files using the column headers.

   The *Data Available* column has a Yes status when there is data for the log file. A No status indicates that no data is available and will result in an empty log file.
10. Select the log file(s) that you want to export or select the checkbox next to *Select All*.
11. Click the *Export* button.
    After the log file is exported, a confirmation message is displayed at the top of the page.

Related Information

Administrator Guide: Repository management, To register a repository in the CMC [page 52]
US Addressing Report [page 2003]
Designer Guide: USPS DSF2 [page 640]
Reference Guide: Transforms, DSF2 Walk Sequencer [page 1172]

4.3.3 Central Repository management

This section describes how to manage your secure central repositories using the Administrator.

When you log into the Management Console as a user with the appropriate rights, the name of each secure central repository appears under the *Central Repositories* node. Links under this node include:

- Users and groups
  Use to add, remove, and configure users and groups for secure object access.
- Reports
  Use to generate reports for central repository objects such as viewing the change history of an object.

> ℹ Note
>
> Before you can manage a secure central repository, it must be registered in the Central Management Console (CMC) and have appropriate user access applied.

Related Information

Administrator Guide: To register a repository in the CMC [page 52]

4.3.3.1 Setting up users and groups

The general process for setting up secure central repository users and groups is as follows:

1. Register the secure central repository in the Central Management Console (CMC).
2. Add central repository groups.
3. Associate users with groups.

The following sections describe these procedures.

Related Information

Designer Guide: Implementing Central Repository Security [page 805]

4.3.3.1.1 Adding a group to a central repository

Groups are specific to a secure central repository and are not visible in any other local or central repository.

1. Expand the *Central Repositories* node in the navigation tree and expand the repository to configure.
2. Click *Users and Groups*.
   The Groups and Users page displays.
3. On the Groups tab, click *Add*.
4. Type a *Name* for the group.
5. Optionally, type a *Description* for the group.
6. Click *Apply*.

The group appears on the Groups tab.

4.3.3.1.2 Adding users

1. Expand the *Central Repositories* node in the navigation tree and expand the repository to configure.
2. Click *Users and Groups*.
   The Groups and Users page displays.
3. Click the *Users* tab.
4. Click *Add*.

   On the Add/Edit User page, enter the following information.

Option	Description
User name	Select the user to add to the group. ℹ Note The list of available users includes all users defined in the Central Management Console (CMC).
Default group	The default central repository group to which the user belongs. You can change the default by selecting another from the drop-down list.
Status	Select a value from the drop-down list: ○ *Active* Enables the user's account for normal activities. ○ *Suspended* Select to disable the login for that user.
Description	Optionally, type a description for the user.

   The User is a member of list on the left shows the groups to which this user belongs.
5. Click *Apply*.

   Clicking *Reset* returns all fields to the last set of values applied.

4.3.3.1.3 To add or remove a user from a group

1. Expand the *Central Repositories* node in the navigation tree and expand the repository to configure.
2. Click *Users and Groups*.
3. Click the *Group* tab.
4. Click the group name.
5. The Members list on the left shows the users in this group.

   To add users to a group, click the user names from the Not Members list and click *Add Users*. Select multiple user names using the Ctrl or Shift keys.

   To remove a user from the group, select a user name from the Members list and click *Remove Users*. Select multiple user names using the Ctrl or Shift keys.

6. Click *Apply*.

   Clicking *Reset* returns all fields to the last set of values applied.

Alternately, click the *Users* tab, click the user name, and associate the user with one or more groups by selecting group names and adding or removing them.

Related Information

Designer Guide: Implementing Central Repository Security [page 805]

4.3.3.1.4 Deleting a group

1. Expand the *Central Repositories* node in the navigation tree, expand the repository to configure, and click *Users and Groups*.
2. Click the *Group* tab.
3. Select the check box for the group.
4. Click *Remove*.

> Note
>
> You cannot delete a group in the following instances:
>
> - It is the default group for any user (whether or not they are active).
> - It is the only group with full permissions for an object.
> - A member of the group is undertaking any central repository tasks using the Designer.

4.3.3.2 Viewing reports

You can generate reports about objects in a central repository such as which objects a user currently has checked out or the changes made to an object over a specified time frame.

Expand the central repository to view and expand the *Reports* link.

Related Information

Object state report [page 1884]

Change report [page 1884]

4.3.3.2.1 Object state report

Use the object state report to view details on one or more objects such as whether the objects are checked out and by whom.

Click the *Object State Report* link to display a search page with the following criteria (all fields are optional):

Option	Description
Object name	Type an object name. You can use the % symbol as a wildcard.
Object type	For example select *Batch job*, *Table*, or *Stored procedure*.
State	For example select *Checked out*.
User	Select a central repository user name.

Click *Search* to generate the report. The report has the following columns:

- *Object name*
- *Object type*
- *State*
- *User name*—The user account associated with the check-out or check-in.
- *Associated repository*—The repository to which the object belongs.
- *Time*—Check-out or check-in date and time.
- *Comments*—Comments added when user checked out or checked in the object.

Click the object name to display the object's history.

Related Information

Designer Guide: Viewing object history [page 822]

4.3.3.2.2 Change report

Use the change report to view the change history for an object over a specified period of time.

Click the *Change Report* link to display a search page with the following criteria:

- *Start date*—Enter a date or click the calendar icon to select a start date.
- *End date*—Enter a date or click the calendar icon to select an end date.
- *Object type*—Optionally select an object type; for example batch job, table, or stored procedure.
- *State*—Optionally select an object state; for example *Checked out*.
- *User*—Optionally select a central repository user name.

Click *Search* to generate the report. The report has the following columns:

- *Object name*
- *Object type*

- *State*
- *Version*—The version number of the object.
- *User name*—The user account associated with the check-out or check-in.
- *Associated repository*—The repository to which the object belongs.
- *Time*—Check-out or check-in date and time.
- *Comments*—Comments added when user checked out or checked in the object.

4.3.4 Server Groups

About this section

Use the Administrator to create and maintain server groups.

This section describes how to work with server groups.

Related Information

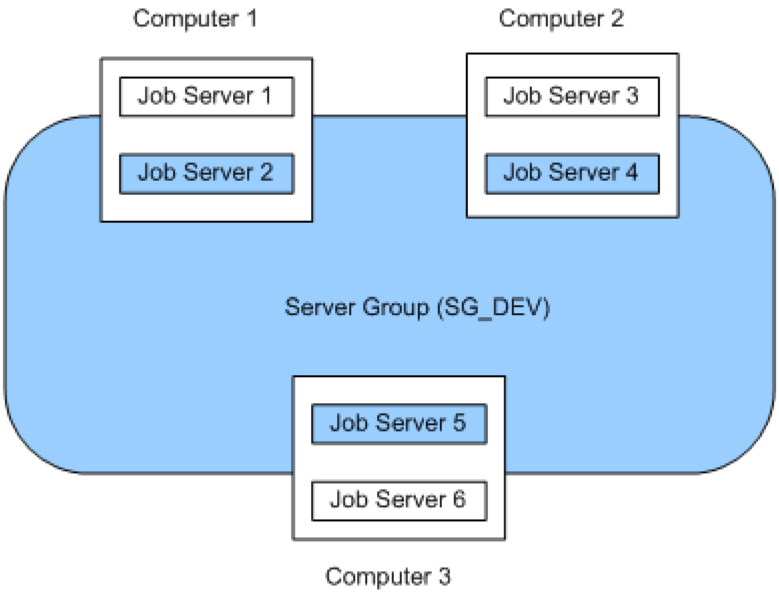
Server group architecture [page 1885]
Adding a server group [page 1888]
Editing and removing a server group [page 1889]
Monitoring Job Server status in a server group [page 1889]
Executing jobs using server groups [page 1890]

4.3.4.1 Server group architecture

You can group Job Servers on different computers into a logical SAP Data Services component called a server group. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at runtime.

There are two rules for creating server groups:

- All the Job Servers in an individual server group must be associated with the same repository, which must be defined as a default repository. The Job Servers in the server group must also have:
  - Identical SAP Data Services versions
  - Identical database server versions
  - Identical locale
- Each computer can only contribute one Job Server to a server group.



The requirement that all Job Servers in a server group be associated with the same repository simply allows you to more easily track which jobs are associated with a server group. It is recommended that you use a naming convention for server groups that includes the name of the repository. For example, for a repository called DEV, a server group might be called `SG_DEV`.

On startup, all Job Servers check the repository to find out if they must start as part of a server group.

Compared to normal Job Servers, Job Servers in a server group each:

- Collect a list of other Job Servers in their server group
- Collect system load statistics every 60 seconds:
  - Number of CPUs (on startup only)
  - Average CPU load
  - Available virtual memory
- Service requests for system load statistics
- Accept server group execution requests

4.3.4.1.1 Load balance index

All Job Servers in a server group collect and consolidate system load statistics and convert them into a load balance index value for each Job Server. A Job Server's load balance index value allows the software to normalize statistics taken from different platforms. The Job Server with the lowest index value is selected to execute the current job. The software polls all Job Server computers every 60 seconds to refresh the load balance index.

4.3.4.1.2 Job execution

After you create a server group, you can select a server group to execute a job from the Designer's Execution Properties window or from the Execute Batch Job, Schedule Batch Job, and Export Batch Job pages in the Administrator.

When you execute a job using a server group, the server group executes the job on the Job Server in the group that is running on the computer that has the lightest load. The Administrator will also resynchronize a Job Server with its repository if there are changes made to the server group configuration settings.

You can execute parts of your job on different Job Servers in a server group. You can select the following distribution levels from the Designer's *Execution Properties* window or from the Execute Batch Job, Schedule Batch Job, and Export Execution Command pages in the Administrator:

- Job level
  A job can execute on an available Job Server.
- Data flow level
  Each data flow within a job can execute on an available Job Server.
- Sub data flow level
  A resource-intensive operation (such as a sort, table comparison, or table lookup) within a data flow can execute on an available Job Server.

Related Information

Performance Optimization Guide: Using grid computing to distribute data flows execution [page 2160]

4.3.4.1.3 Job launcher

The Job Launcher, exported as part of a job's execution commands, includes a specific command line option for server groups. You can use this option to change the Job Servers in a server group.

Related Information

About the job launcher [page 1903]

4.3.4.1.4 Working with server groups and Designer options

Some Designer options assume paths are relative to a Job Server. If your Job Servers are on different machines from your Designer (typically the case in a production environment) you must ensure that connections and directory paths point to the Job Server host that will run the job. Such options include:

- Source and target directories for files
- Bulk load directories
- Source and target connection strings to databases
- Path to repositories

When using server groups consider the additional layer of complexity for connections. For example, if you have three Job Servers in a server group:

- Use the same directory structure across your three host computers for source and target file operations and use relative paths for file names.
- Use the same connection strings to your databases for all three Job Server hosts.

If you use job distribution levels, the Job Servers in the server group must have:

- Identical SAP Data Services versions
- Identical database server versions
- Identical locale
- Identical operating systems

Thoroughly test the Job Server job options when working with server groups.

Adding a server group:

- In the Administrator, use the Server Groups node to create and add a server group.

4.3.4.1.4.1 Adding a server group

1. Select *Server Groups* > *All Server Groups* .
2. Click the *Server Group Configuration* tab.
3. Click *Add*.
4. Follow the instructions on the Add Server Group page to create a server group.
   - When you select a repository, all Job Servers registered with that repository display. You can create one server group per repository.
   - Notice that the Administrator provides a default server group name. It is the name of your repository with the prefix `SG_` (for server group). You can change the default name, however, labeling a server group with the repository name is recommended.
   - One Job Server on a computer can be added to a server group. Use the *Host and Port* column to verify that the Job Servers you select are each installed on a different host.
5. After you select the Job Servers for a server group, click *Apply*.

   The display returns to the Server Group Configuration page.

Related Information

Monitoring Job Server status in a server group [page 1889]

4.3.4.2 Editing and removing a server group

You can select a new set of Job Servers for an existing server group or remove a server group.

Trace messages are written for a change in Job Server status when you create, edit, or remove server groups.

- When a Job Server is upgraded to membership in a server group, the trace message is:
  `Collecting system load statistics, maintaining list of Job Server(s) for this server group, and accepting Job Server execution requests.`
- When a Job Server is downgraded out of a server group, the trace message is:
  `Deleting current system load statistics, and not collecting more. Not accepting job execution requests from a server group.`

4.3.4.2.1 Editing a server group

1. Select a server group from the navigation pane on the left.
2. In the *Server Groups* page, click the *Server Group Configuration* tab.
3. Select a new set of Job Servers.
4. Click *Apply*.

   Your edited server group is saved and the display returns to the *Server Groups Configuration* tab.

4.3.4.2.2 Removing a server group

1. In the *Server Groups* page, click the *Server Group Configuration* tab.
2. Select the check box for the server group(s) that you want to remove.
3. Click *Remove*.

> **Note**
>
> If you delete Job Servers from a repository in order to delete all the Job Servers in a server group, the Administrator displays an invalid status for the server group.

4.3.4.3 Monitoring Job Server status in a server group

If Job Servers are in a server group, you can view their status in the Administrator.

- To monitor the status of these Job Servers, select *Server Groups* > *All Server Groups*.
  The Server Group Status page opens. All existing server groups are displayed with the Job Servers they contain.

Indicator	Description
✅	A green icon indicates that a Job Server is running.
⚠	A yellow icon indicates that a Job Server is not running.
❌	A red icon indicates that the Job Server cannot connect to the repository.

If a server group contains Job Servers with a mix of green, yellow, or red indicators, then its indicator appears yellow:
Otherwise, a server group indicator displays the same color indicator as its Job Servers.

- To view the status for a single server group, select its name.

4.3.4.4 Executing jobs using server groups

After you create a server group, you can select a server group to execute a job from the Designer's *Execution Properties* window or from the Execute Batch Job and Schedule Batch Job pages in the Administrator.

Related Information

Batch Jobs [page 1890]

4.3.5 Batch Jobs

About this section

This section describes how to execute, schedule, and monitor batch jobs from the Administrator.

Before you can manage batch jobs with the Administrator, add repository connections.

4.3.5.1 Executing batch jobs

You can execute batch jobs from the Administrator if their repositories are registered in the Central Management Console (CMC) and your user has the appropriate rights.

1. Select *Batch* > **`<repository>`**.

   The Administrator opens the Batch Job Status page, which lists all of the jobs in the selected repository.

   To view jobs in all repositories from this page, select *Batch* > *All Repositories*. (The All Repositories option appears under the Batch Job node if more than one repository is connected to the Administrator.)

2. Click the *Batch Job Configuration* tab.
3. To the right of the job you want to run, click *Execute*.

   The Administrator opens the Execute Batch Job page.
4. Under *Enter Execution Options*, set the parameters for the execution of this job.
5. Under *Select Trace Options,* set the trace properties for this execution of the job.
6. Click *Execute* to run the job.

   The Administrator returns to the Batch Job Status page.

Related Information

Reference Guide: Objects, Batch Job, Parameters [page 840]
Reference Guide: Objects, Batch Job, Trace properties [page 843]

4.3.5.2 Scheduling jobs

There are three ways to manage job schedules.

Related Information

Using the job scheduler [page 1891]
Scheduling jobs in SAP BusinessObjects Business Intelligence platform [page 1897]
Using a third-party scheduler [page 1899]

4.3.5.2.1 Using the job scheduler

When you schedule batch jobs using the SAP Data Services job scheduler, it creates an entry in the operating system's scheduling utility on the Job Server computer. Windows uses the Task Scheduler and UNIX systems use the CRON utility. (Note that if you make changes to a schedule directly through these utilities, the job scheduler will not reflect those changes.)

Related Information

Adding a job schedule [page 1892]
Activating or deactivating one or more job schedules [page 1895]
Updating a job schedule [page 1895]

Removing a job schedule [page 1896]
Migration considerations [page 1896]

4.3.5.2.1.1 Adding a job schedule

1. Select *Batch* > **<repository>**.
2. Click the *Batch Job Configuration* tab.
3. For the job to configure, click *Add Schedule*.
4. On the *Schedule Batch Job* page, enter the desired options:

Option	Description
Enter a job schedule	
Schedule name	Enter a unique name that describes this schedule. Note You cannot rename a schedule after you create it.
Active	Select this box to enable (activate) this schedule, then click *Apply*. This option allows you to create several schedules for a job and then activate the one(s) you want to run.
Select a scheduler	
Data Services scheduler	Creates the schedule on the Job Server computer.
BOE scheduler	Creates the schedule on the selected Central Management Server (CMS).
Select scheduled day(s) for executing the job	
Calendar	From the drop-down list on the calendar, select: ○ *Day of Week* to schedule the job by the day of the week. You can select one or more days. Click again to deselect. ○ *Day of Month* to schedule the job by date. You can select one or more dates. Click again to deselect. If *Recurring* is selected, then the Administrator schedules this job to repeat every week or month on the selected day. Note that if you select multiple days of the week or month, the job will run on a recurring basis by default.
Select scheduled time for executing the jobs	
Once a day	Enter the time for the scheduler to start the job (hours, minutes, and either AM or PM).
Multiple times a day	○ For the Data Services scheduler, enter the time (hours, minutes, and either AM or PM) for the scheduler to repeatedly run the job

Option	Description
	for the selected duration (in minutes) at the selected interval (in minutes). ○ For the BOE scheduler, enter (in minutes) the repeat interval to run the job. You must also select all days in the calendar (for weekly or monthly). Select a time when all of the required resources are available. Typically, you want to schedule jobs to ensure they finish before the target database or data warehouse must be available to meet increased demand.
Select job execution parameters	
System configuration	Select the system configuration to use when executing this job. A system configuration defines a set of datastore configurations, which define the datastore connections. For more information, see “Creating and managing multiple datastore configurations” in the *Designer Guide*. If a system configuration is not specified, the software uses the default datastore configuration for each datastore. This option is a run-time property. This option is only available if there are system configurations defined in the repository.
Job Server or server group	Select the Job Server or a server group to execute this schedule.
Use password file	Select to create or update the password file that the job schedule accesses for current repository connection information. Deselect the option to generate the batch file with a hard-coded repository information. Note This option is disabled if you have not set up a CMS connection.
Enable auditing	Select this option if you want to collect audit statistics for this specific job execution. This option is selected by default. For more information about auditing, see “Using Auditing” in the *Designer Guide*.
Disable data validation statistics collection	Select this option if you do not want to collect data validation statistics for any validation transforms in this job. This option is not selected by default.
Enable recovery	Select this option to enable the recovery mode when this job runs.
Recover from last failed execution	Select this option if an execution of this job has failed and you want to enable the recovery mode.

Option	Description
Collect statistics for optimization	Select this option to collect statistics that the optimizer will use to choose an optimal cache type (in-memory or pageable). This option is not selected by default. See "Using statistics for cache self-tuning" in the *Performance Optimization Guide*.
Collect statistics for monitoring	Select this option to display cache statistics in the Performance Monitor in the Administrator. This option is not selected by default. See "Monitoring and tuning cache types" in the *Performance Optimization Guide*.
Use collected statistics	Select this option if you want the optimizer to use the cache statistics collected on a previous execution of the job. The option is selected by default. For more information, see "Using statistics for cache self-tuning" in the *Performance Optimization Guide*.
Export Data Quality reports	Generates and exports all specified job reports to the location specified in the *Management* ▸ *Report Server Configuration* node. By default, the reports are exported to **`<DS_COMMON_DIR>`**`\DataQuality\reports\`**`<repository\job>`**.
Distribution level	Select the level within a job that you want to distribute to multiple Job Servers for processing: ○ *Job*: The whole job will execute on an available Job Server. ○ *Data flow*: Each data flow within the job can execute on an available Job Server. ○ *Sub data flow*: Each sub data flow (can be a separate transform or function) within a data flow can execute on an available Job Server. For more information, see "Using grid computing to distribute data flows execution" in the *Performance Optimization Guide*.

5. Click *Apply*. Clicking *Reset* returns all fields to the last set of values applied.

Related Information

Designer Guide: Datastores, Creating and managing multiple datastore configurations [page 239]
Managing database account changes [page 1873]
Reference Guide: Objects, Batch Job, Parameters [page 840]
Designer Guide: Data Assessment, Using Auditing [page 452]
Data Validation Dashboard Reports [page 1981]
Configuring the report server [page 1875]
Performance Optimization Guide: Using Caches, Using statistics for cache self-tuning [page 2135]

Performance Optimization Guide: Using Caches, To monitor and tune in-memory and pageable caches [page 2135]

Performance Optimization Guide: Distributing Data Flow Execution, Using grid computing to distribute data flows execution [page 2160]

4.3.5.2.1.1.1 Activating or deactivating one or more job schedules

In order for a job schedule to run, it must be active.

To change an existing job schedule, you must first deactivate it, make the changes, then reactivate it.

1. Select *Batch* **`<repository>`** .
2. Click the *Repository Schedules* tab.

   In order for a job schedule to run, it must be active.

   The Repository Schedules tab lists all schedules for all jobs in the repository, and you can remove, activate, or deactivate one or more schedules.

   Alternately, click the Batch Job Configuration tab, then for a particular job, click the Schedules link. The Batch Job Schedules tab lists all schedules for that particular job. Here you can add, remove, activate, or deactivate one or more schedules:

   The Job Server column listed next to each schedule indicates which Job Server will execute it.

   If there is a server group icon in the Job Server column, this indicates the schedule will be executed by the server group, and the schedule is stored on the indicated Job Server. To see which server group is associated with the schedule, roll your cursor over the server group icon.

   If there is CMS icon in the Job Server column, this indicates the job schedule is managed by a Central Management Server.

   Click the System Configuration names, if configured, to open a page that lists the datastore configurations in that system configuration.
3. On either the *Repository Schedules* tab or the *Batch Job Schedules* tab, select one or more check boxes for a schedule.
4. Click *Activate* (or *Deactivate*).

Related Information

Updating a job schedule [page 1895]

4.3.5.2.1.2 Updating a job schedule

To edit a job schedule, you must first deactivate it, make the changes, then reactivate it.

1. Select *Batch* **<repository>**.
2. Click the *Batch Job Configuration* tab.
3. Click the *Schedules* link for the desired job.
4. Click the schedule name to edit.
5. The *Schedule Batch Job* page displays.
6. If the schedule is currently active, deactivate it by clearing the *Active* check box and click *Apply*.

   > Note
   >
   > You do not need to deactivate the schedule to update most of the job execution parameters at the bottom of the page. Only the schedule-related parameters require deactivation in order to update them.

7. Edit the schedule parameters as required.
8. To reactivate the schedule now, select the *Active* check box.
9. Click *Apply*.

   The status bar at the top of the page confirms that the schedule has been created and/or activated.

Related Information

Adding a job schedule [page 1892]

4.3.5.2.1.3 Removing a job schedule

1. Select *Batch* **<repository>**.
2. Click the *Repository Schedules* tab.
3. Select one or more check boxes for a schedule.
4. Click *Remove*.

   The Administrator deletes the information about this job schedule.

4.3.5.2.1.4 Migration considerations

Changes made to the Job Server, such as an upgrade, do not affect schedules created in SAP Data Services as long as:

- The new version of the software is installed in the same directory as the original version (Data Services schedulers use a hard-coded path to the Job Server).
- The new installation uses the Job Server name and port from the previous installation. (This occurs automatically when you install over the existing `DSConfig.txt` file.)

When you export a repository via an `.atl` file, jobs and their schedules (created in Data Services) automatically export as well.

You can also import a repository `.atl` file including jobs and their associated schedules (previously created in Data Services) back into Data Services.

Remember that once imported, you must reactivate job schedules to use them. If the job schedule uses a password file, then reactivating it will automatically generate the password file.

Related Information

Administrator Guide: Importing from a File [page 122]

4.3.5.2.2 Scheduling jobs in SAP BusinessObjects Business Intelligence platform

If you are using SAP BusinessObjects Business Intelligence platform and you want to manage your SAP Data Services job schedules in that application, first create a connection to a Central Management Server (CMS), then configure the schedule to use that server.

4.3.5.2.2.1 Adding a CMS connection

1. Select ▶ *Management* ▶ *CMS Connection* ◀.
2. Click *Add*.
3. On the *CMS Connections* page, enter the connection information.

   The parameters in the top section are the same as when logging in to an SAP BusinessObjects Business Intelligence platform Central Management Console (CMC) or InfoView. For details, refer to the *SAP BusinessObjects Business Intelligence platform InfoView User's Guide*.

   The parameters in the bottom section (User account credentials for executing the program) depend on how the CMS server is set up. For details, refer to "Authentication and program objects" in the *SAP BusinessObjects Business Intelligence platform Administrator Guide*.

Option	Description
System	Type the computer name that hosts the Central Management Server (CMS), a colon, and the port number.
User name	Type the CMC/InfoView user name.
Password	Type the CMC/InfoView user password.
Authentication	Select the authentication type for the server
User account credentials for executing the program (optional)	

Option	Description
Note If you do not have the following option cleared in the Central Management Console, you will be required to enter user account credentials in order for your schedules to run: In the CMC, select *Objects* > *Objects Settings* > *Program objects* and clear the *Use Impersonation* option.	
User name	The CMS computer might require operating system login credentials to run the schedule. If so, type the user name (and password) for the applicable account.
Password	The CMS computer might require operating system login credentials to run the schedule. If so, type the (user name and) password for the applicable account.

4. Click *Apply*.

4.3.5.2.2.2 Creating a job schedule in SAP BusinessObjects Business Intelligence platform

1. Select *Batch* > **`<repository>`**.
2. Click the *Repository Schedules* tab.
3. Click the schedule name to configure.
4. If the schedule is currently active, deactivate it by clearing the *Active* check box and click *Apply*.
5. Edit the schedule parameters as necessary.

> Note
>
> Time-sensitive parameters reflect the time zone of the computer where the Administrator is installed, not where the CMS is installed.

6. Under the *Select a scheduler* section, select *BOE scheduler*.
7. From the drop-down list, select a CMS name.
8. To reactivate the schedule now, select the *Active* check box.
9. Click *Apply*.

   The status bar at the top of the page confirms that the schedule has been created and/or activated.

   If it doesn't already exist, SAP BusinessObjects Business Intelligence platform creates a folder called Data Services and stores the schedule file and a parameters file (called schedulename.txt).

   For a BOE schedule with the *Use password file* option selected, then SAP Data Services also creates a password file in the Data Services folder (called `repositoryname.txt`)

Note

When you deactivate a schedule created on a CMS, SAP BusinessObjects Business Intelligence platform deletes the object. Therefore, any changes made to the calendar will be lost.

4.3.5.2.2.3 Removing a CMS connection

1. Select *Management* > *CMS Connection* .
2. Select the check box for the connection to remove from the administrator.
3. Click *Remove*.

4.3.5.2.3 Using a third-party scheduler

When you schedule jobs using third-party software:

- The job initiates outside of SAP Data Services.
- The job runs from an executable batch file (or shell script for UNIX) exported from Data Services.
- When you execute a Data Services job using the exported execution command on a Unix/Linux environment from a third-party scheduler or any other application, source the `$LINK_DIR/bin/al_env.sh` file before calling the script that contains the command to start the job. This can be done by adding following lines in the beginning of the script containing the execution command:

  ```
  export LINK_DIR=</usr/sw/sap/dataservices>
  . $LINK_DIR/bin/al_env.sh
  ```

  Where /usr/sw/sap/dataservices is the directory where Data Services is installed.

Note

When a third-party scheduler invokes a job, the corresponding Job Server must be running.

Related Information

About the job launcher [page 1903]

4.3.5.2.3.1 Executing a job with a third-party scheduler

1. Export the job's execution command to an executable batch file (`.bat` file for Windows or `.sh` file for UNIX environments).

2. Ensure that the Data Services Service is running (for that job's Job Server) when the job begins to execute.

   The Data Services Service automatically starts the Job Server when you restart the computer on which you installed the Job Server.

   ○ You can also verify whether a Job Server is running at any given time using the Designer. Log in to the repository that contains your job and view the Designer's status bar to verify that the Job Server connected to this repository is running.
   ○ You can verify whether all Job Servers in a server group are running using the Administrator. In the navigation tree select *Server Groups* > *All Server Groups* to view the status of server groups and the Job Servers they contain.
3. Schedule the batch file from the third-party software.

> i Note
>
> To stop an SAP Data Services job launched by a third-party scheduling application, press CTRL+C on the application's keyboard.

4.3.5.2.3.2 To export a job for scheduling

1. Select *Batch* > **`<repository>`**.
2. Click the *Batch Job Configuration* tab.
3. For the batch job to configure, click the *Export Execution Command* link.
4. On the *Export Execution Command* page, enter the desired options for the batch job command file that you want the Administrator to create:

Option	Description
File name	The name of the batch file or script containing the job. The third-party scheduler executes this file. The Administrator automatically appends the appropriate extension: ○ `.sh` for UNIX ○ `.bat` for Windows
System configuration	Select the system configuration to use when executing this job. A system configuration defines a set of datastore configurations, which define the datastore connections. For more information, see “Creating and managing multiple datastore configurations” in the *Designer Guide*. If a system configuration is not specified, the software uses the default datastore configuration for each datastore. This option is a run-time property. This option is only available if there are system configurations defined in the repository.
Job Server or server group	Select the Job Server or a server group to execute this schedule.

Option	Description
Enable auditing	Select this option if you want to collect audit statistics for this specific job execution. The option is selected by default. For more information about auditing, see “Using Auditing” in the *Designer Guide*.
Disable data validation statistics collection	Select this option if you do not want to collect data validation statistics for any validation transforms in this job. The option is not selected by default.
Enable recovery	Select this option to enable the automatic recovery feature. When enabled, the software saves the results from completed steps and allows you to resume failed jobs. See “Automatically recovering jobs” in the *Designer Guide* for information about the recovery options.
Recover from last failed execution	Select this option to resume a failed job. The software retrieves the results from any steps that were previously executed successfully and re-executes any other steps. This option is a run-time property. This option is not available when a job has not yet been executed or when recovery mode was disabled during the previous run.
Use password file	Select to create or update a password file that automatically updates job schedules after changes in database or repository parameters. Deselect the option to generate the batch file with a hard-coded repository user name and password. Note This option is disabled if you have not set up a CMS connection.
Collect statistics for optimization	Select this option to collect statistics that the optimizer will use to choose an optimal cache type (in-memory or pageable). This option is not selected by default. See “Using statistics for cache self-tuning” in the *Performance Optimization Guide*.
Collect statistics for monitoring	Select this option to display cache statistics in the Performance Monitor in the Administrator. The option is not selected by default. For more information, see “Monitoring and tuning cache types” in the *Performance Optimization Guide*.
Use collected statistics	Select this check box if you want the optimizer to use the cache statistics collected on a previous execution of the job. The option is selected by default. See “Using statistics for cache self-tuning” in the *Performance Optimization Guide*.

Option	Description
Export Data Quality reports	Generates and exports all specified job reports to the location specified in the *Management* > *Report Server Configuration* node. By default, the reports are exported to **<DS_COMMON_DIR>**`\DataQuality\reports\`**<repository\job>**.
Distribution level	Select the level within a job that you want to distribute to multiple Job Servers for processing: ○ *Job*: The whole job will execute on one Job Server. ○ *Data flow*: Each data flow within the job will execute on a separate Job Server. ○ *Sub data flow:* Each sub data flow (can be a separate transform or function) within a data flow can execute on a separate Job Server. For more information, see “Using grid computing to distribute data flows execution” in the *Performance Optimization Guide*.

5. Click *Export*.

   The Administrator creates command files **<filename>**`.txt` (the default for filename is the job name) and a batch file for the job and writes them to the local **<DS_COMMON_DIR>**`\log` directory.

i Note

You can relocate the password file from the **<DS_COMMON_DIR>**`\conf` directory, but you must edit the **<filename>**`.txt` file so that it refers to the new location of the password file. Open the file in a text editor and add the relative or absolute file path to the new location of the password file in the argument `-R "`**<repositoryname>**`.txt"`.

Related Information

Data Validation Dashboard Reports [page 1981]
Managing database account changes [page 1873]
Configuring the report server [page 1875]
Designer Guide: Datastores, Creating and managing multiple datastore configurations [page 239]
Reference Guide: Objects, Batch Job, Execution options [page 840]
Designer Guide: Data Assessment, Using Auditing [page 452]
Performance Optimization Guide: Using Caches, Monitoring and tuning caches [page 2134]
Performance Optimization Guide: Using Caches, Using statistics for cache self-tuning [page 2135]
Performance Optimization Guide: Distributing Data Flow Execution, Using grid computing to distribute data flow execution [page 2160]

4.3.5.2.4 About the job launcher

SAP Data Services exports job execution command files as batch files on Windows or CRON files on UNIX. These files pass parameters and call `AL_RWJobLauncher`. Then, `AL_RWJobLauncher` executes the job, sends it to the appropriate Job Server, and waits for the job to complete.

> ⚠ Caution
>
> Do not modify the exported file without assistance from SAP Business User Support.

Related Information

Administrator Guide: Configuring SSL for Data Services components [page 34]
Administrator Guide: Configuring SSL for the CMS connection [page 41]

4.3.5.2.4.1 Job launcher flag values and arguments

The following table lists job launcher flags and their values.

Flag	Value
`-w`	The job launcher starts the job(s) and then waits before passing back the job status. If -w is not specified, the launcher exits immediately after starting a job.
`-t`	The time, in milliseconds, that the Job Server waits before checking a job's status. This is a companion argument for `-w`.
`-s`	Status or return code. 0 indicates successful completion, non-zero indicates an error condition. Combine `-w`, `-t`, and `-s` to execute the job, wait for completion, and return the status.
`-C`	Name of the engine command file (path to a file which contains the Command line arguments to be sent to the engine).
`-v`	Prints `AL_RWJobLauncher` version number.
`-S`	Lists the server group and Job Servers that it contains using the following syntax: `"SvrGroupName;inet:JobSvr1Name:JobSvr1Host:JobSvr1Port;inet:JobSvr2Name:JobSvr2Host:JobSvr2Port";` For example: `"SG_DEV;inet:HPSVR1:3500;inet:WINSVR4:3505";`
`-R`	The location and name of the password file. Replaces the hard-coded repository connection values for `-S, -N, -U, -P`.

Flag	Value
`-xCR`	Generates and exports all specified job reports to the location specified in the *Management* › *Report Server Configuration* node. By default, the reports are exported to **`<DS_COMMON_DIR>`**`\DataQuality\reports\`**`<repository\job>`**. In order to use this flag, you must disable the security for the Export_DQReport operation in the *Administrator* › *Web Services* › *Web Services Configuration* tab.

There are two arguments that do not use flags:

- `inet` address: The host name and port number of the Job Server. The string must be in quotes. For example:

  ```
  "inet:HPSVR1:3500"
  ```

  If you use a server group, inet addresses are automatically rewritten using the -S flag arguments. On execution, the first Job Server in the group checks with the others and the Job Server with the lightest load executes the job.
- `server log path`: The fully qualified path to the location of the log files. The server log path must be in quotes. The server log path argument does not appear on an exported batch job launch command file. It appears only when Data Services generates a file for an active job schedule and stores it in the following directory: **`<DS_COMMON_DIR>`**`/Log/`**`<JobServerName>`**`/`**`<RepositoryName>`**`/`**`<JobInstanceName>`** . You cannot manually edit server log paths.

Related Information

Integrator Guide: To configure web service information using the Administrator [page 2221]

4.3.5.2.4.2 Job launcher error codes

The job launcher also provides error codes to help debug potential problems. The error messages are:

Error number	Error message
180002	Network failure.
180003	The service that will run the schedule has not started.
180004	`LINK_DIR` is not defined.
180005	The trace message file could not be created.
180006	The error message file could not be created.
180007	The GUID could not be found. The status cannot be returned.
180008	No command line arguments were found.

Error number	Error message
180009	Invalid command line syntax.
180010	Cannot open the command file.

4.3.6 Real-Time Jobs

This section describes how to support real-time jobs using the Administrator.

Before configuring services, add real-time job repository and Access Server connections to the Administrator.

Related Information

Supporting real-time jobs [page 1905]
Configuring and monitoring real-time services [page 1907]
Creating and monitoring client interfaces [page 1915]

4.3.6.1 Supporting real-time jobs

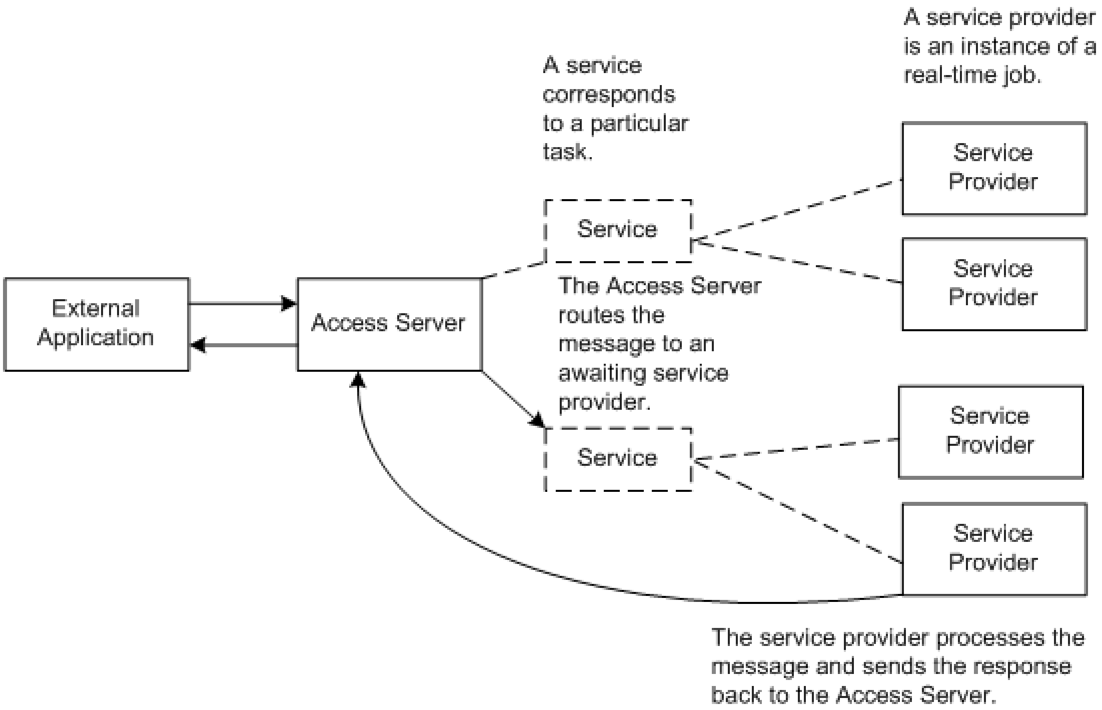
The Access Server manages real-time communication between SAP Data Services and external applications (such as ERP or web applications). The Access Server determines how to process incoming and outgoing messages based on the settings you choose for each real-time job in the Administrator.

In particular you use the Administrator to define:

- Services: A service is a name that identifies a task. The Access Server receives requests for a service. You associate a service with a real-time job. The real-time job contains the real-time processing loop that can process requests for this service and generate a response.
- Service providers: A service provider is the engine process that performs a service; the service provider completes the tasks in a real-time job. A service provider is controlled by a Job Server. A Job Server can control several service providers—each service provider is a unique process instance.

The Access Server uses services and service providers to process message requests. The following is an example:

1. An external application sends a request to the Access Server.
2. The Access Server determines the appropriate service for the request.
3. The Access Server finds the associated service providers and dispatches the request to the next available service provider.
4. Under the control of a Job Server, that service provider completes the processing for the request. A different Job Server might control each service provider.


A service corresponds to a particular task.
A service provider is an instance of a real-time job.
Service Provider
Service Provider
Service
External Application
Access Server
The Access Server routes the message to an awaiting service provider.
Service Provider
Service
Service Provider
The service provider processes the message and sends the response back to the Access Server.

The Access Server manages the entire set of service providers, implementing configuration changes and telling the appropriate Job Servers to start and stop service providers. At a prescribed interval, the Access Server updates service providers, balancing loads and implementing configuration changes. To balance loads, the Access Server monitors requests for services to ensure that no service provider is over-used or under-used. Based on the number of requests for a service, the Access Server tells Job Servers to start or stop service providers.

To support real-time jobs, you must:

- Create any number of Access Servers using the Server Manager utility, then add a connection to each local or remote Access Server using the *Management* node in the Management Console Administrator.
- In the *Real-Time* node of the Administrator, create a service for each real-time job under each Access Server's node.
- Create one or more service providers for each service.
- Start the services.
- Monitor the services.

Related Information

Creating services and service providers [page 1907]
Starting and stopping services [page 1910]
Monitoring services [page 1914]

4.3.6.2 Configuring and monitoring real-time services

To enable an Access Server to support real-time jobs, you must configure and monitor real-time services and service providers for it.

- Configure services by specifying a real-time job and other operational parameters.
- Configure service providers by specifying a Job Server and indicating the maximum and minimum number of instances that you want the Job Server to control. Each service provider is a unique process or instance controlled by a Job Server.

Related Information

Creating services and service providers [page 1907]
Starting and stopping services [page 1910]
Updating service providers [page 1913]
Monitoring services [page 1914]

4.3.6.2.1 Creating services and service providers

In the Administrator, you create a service that processes requests for each real-time job. You also create the service providers to perform that service. A service provider is the process that completes the tasks in a real-time job.

4.3.6.2.1.1 Adding a service

1. Select ▶ *Real-time* ▶ **`<Access Server>`** ▶ *Real-Time Services* ◀.
2. Click the *Real-Time Services Configuration* tab.
3. Click the *Add* button.
4. In the *Service configuration* section, enter information that describes this service.

Parameter	Description
Service name	A unique name for this service.
Job name	Click *Browse Jobs* to view a list of all the real-time jobs available in the repositories that you connected to the Administrator. Select a job name to fill the service configuration form.
Repository name	Logical name for a repository (used in the Administrator only).

Parameter	Description
Processing retry count max	The number of times that the Access Server attempts to restart a job that fails to respond.
Enable job tracing	A flag that indicates whether the service will write trace messages. Select *Enable* for the job to write trace messages.
Startup timeout	The maximum time that the Access Server waits for the service to register after startup (in seconds).
Queuing timeout	The maximum time that the Access Server waits for the service to process the request (in seconds).
Processing timeout	The maximum time that the Access Server waits for a response from the service (in seconds).
Recycle request count max	The number of requests that the Access Server sends to a given real-time service before automatically recycling the flow.
System Configuration	If configured, select the system configuration to use when executing this service. This option is available only if there are system configurations defined in the repository. For more information, see "Parameters" in the *Reference Guide*.
Enable	A flag that indicates whether the Access Server attempts to automatically start this service when the Access Server restarts. Select *Enable* if you want to automatically start this service when the Access Server restarts. This is the default setting. If you clear the *Enable* option, when the Access Server restarts, it does not automatically start this service. If you manually attempt to start a disabled service, an error message appears in the Service's *Status* column.

5. Add one or more job servers to start this service provider. In the *Service provider* section, click the *Add* button to insert a new Job Server.
6. In the *Job Server* list, select a Job Server to control the service provider. Job Servers are defined by host name and port number.

   You may not select a Job Server more than one time.
7. In the *Min instances* and *Max instances* fields, enter a minimum and a maximum number of service providers that you want this Job Server to control for this service.
8. The job server is enabled by default. To configure but not start the service providers controlled by this Job Server, select the checkbox next to the Job Server, and click the *Disable* button.
9. To add a substitution parameter, click the *Add Overridden Parameter* link.
10. From the drop-down list, select the substitution parameter to override, and enter the override value.
11. Click *Apply*.

The Administrator updates the configuration parameters for this service. These configuration parameters apply to all providers of this service.

When you are ready for the Access Server to process requests, start the service.

Related Information

Service startup behavior [page 1923]
High-traffic behavior [page 1924]
Response time controls [page 1925]
Starting a service [page 1910]
Reference Guide: Objects, Batch Job, Parameters [page 840]

4.3.6.2.1.2 Adding or changing a service provider for a service

1. Select ▶ *Real-time* ▶ `<Access Server>` ▶ *Real-Time Services* ◀.
2. Click the *Real-Time Services Configuration* tab.
3. Click the name of the service for which you want to change the service provider.
4. In the *Service provider* section, click the *Add* button.
5. In the *Job Server* list, select a Job Server to control the service provider. Job Servers are defined by host name and port number.

   You may not select a Job Server more than one time.
6. In the *Min instances* and *Max instances* fields, enter a minimum and a maximum number of service provider instances that you want this Job Server to control.
7. The job server is enabled by default. To configure but not start the service providers controlled by this Job Server, select the checkbox next to the Job Server, and click the *Disable* button.
8. Click the *Apply* button.

   If the service has already started, the Access Server adds this service provider to the available list when it next updates the service providers.

   If the service has not yet started, the Access Server starts enabled service providers when the service starts.

Related Information

Updating service providers [page 1913]

4.3.6.2.1.3 Setting the service provider update interval

The *Provider update interval for services* option sets the time interval, in seconds, between service provider updates. Valid values range from 10 to 120 seconds. The default is 30 seconds. When updating service providers, the Access Server balances loads and implements any configuration changes you have applied to a service provider.

If the provider update interval is too small, performance can decrease because the Access Server must frequently check for events and collect statistics. It's recommended that you set the Provider update interval for services to 30 seconds. On systems with heavy loads and production systems with fewer start and stop events, you can increase the interval.

1. Select ▶ *Real-Time* > `<Access Server>` > *Status* ◢.

   This is the time interval, in seconds, between service provider updates. Valid values range from 10 to 120 seconds. The default is 30 seconds. When updating service providers, the Access Server balances loads and implements any configuration changes you have applied to a service provider.

   If the provider update interval is too small, performance can decrease because the Access Server must frequently check for events and collect statistics. It's recommended that you set the *Provider update interval for services* option to 30 seconds. On systems with heavy loads and production systems with fewer start and stop events, you can increase the interval.
2. Click the *Access Server Configuration* tab.
3. Enter the desired *Provider update interval for services*.

Related Information

Updating service providers [page 1913]

4.3.6.2.2 Starting and stopping services

After you create the required services and service providers, you can start them. After you start a service or service provider, SAP Data Services ensures that it continues to run. You can also use the Administrator to stop a service (such as for maintenance). Similarly, use the Administrator to remove, enable, or disable services and service providers.

4.3.6.2.2.1 Starting a service

1. Select ▶ *Real-Time* > `<Access Server>` > *Real-Time Services* ◢.
2. In the Real-Time Services Status tab, select the check box next to the service or services that you want to start.
3. Click *Start*.

The Access Server starts the minimum number of service providers for this service.

4.3.6.2.2.2 Enabling a service

1. Select ▶ *Real-Time* ▶ **`<Access Server>`** ▶ *Real-Time Services* ◢.
2. Click the *Real-Time Services Configuration* tab. The table of services displays the status in the Enabled column.
3. Select the check box next to the service or services that you want to enable.
4. Click the *Enable* button.

   The Access Server enables the minimum number of service providers for this service, and the *Enabled* column displays Yes.

This change does not start the service. Instead, the service is enabled the next time that the Access Server attempts to start the service, such as after the Access Server restarts.

4.3.6.2.2.3 Disabling a service

1. Select ▶ *Real-Time* ▶ **`<Access Server>`** ▶ *Real-Time Services* ◢.
2. Click the *Real-Time Services Configuration* tab. The table of services displays the status in the Enabled column.
3. Select the check box next to the service or services that you want to disable.
4. Click the *Disable* button.

This change does not have an immediate effect on the service. Instead, the service is disabled the next time that the Access Server attempts to start the service, such as after the Access Server restarts. If you attempt to start a disabled real-time service, an error message displays in the service's status.

4.3.6.2.2.4 To abort or shut down a service

1. Select ▶ *Real-Time* ▶ **`<Access Server>`** ▶ *Real-Time Services* ◢.
2. In the Real-Time Services Status tab, select the check box next to the service or services that you want to abort or shut down.

Option	Description
Abort	Shuts down all service providers for this service without waiting for them to complete processing. The Access Server responds to current and new requests for this service with an error.
Shutdown	Shuts down all service providers for this service after they complete processing any current requests. The Access Server responds to new requests for this service with an error.

3. Click *Abort* or *Shutdown*.

Related Information

Starting a service [page 1910]
Disabling a service [page 1911]

4.3.6.2.2.5 Removing a service

1. Select ▶ *Real-Time* ▶ **`<Access Server>`** ▶ *Real-Time Services* ▶.
2. Click the *Real-Time Services Configuration* tab.
3. Select the check box next to the service or services that you want to remove.
4. Click *Remove*.

   The Administrator stops processing this service. The Access Server shuts down each of the service providers defined for this service and removes the service from the list.

4.3.6.2.2.6 To remove, enable, or disable a service provider

1. Select ▶ *Real-Time* ▶ **`<Access Server>`** ▶ *Real-Time Services* ▶.
2. Click the *Real-Time Services Configuration* tab.
3. If you want to change a Job Service's service providers, select the check box next to a Job Server.
4. Click one of the buttons below the list of Job Servers to perform the appropriate action:

Option	Description
Enable	Start the service providers controlled by the selected Job Servers. Each Job Server starts the minimum number of service providers. The Access Server now includes the selected Job Servers in the set of available service providers. If a Job Server is already enabled, this choice has no effect.
Remove	Discontinue using the service providers controlled by the selected Job Servers to process requests for this service. The Access Server shuts down the service providers and removes the Job Server from the list.

5. Shut down the service providers controlled by the selected Job Servers. The Access Server finishes processing any current requests and no longer includes the selected Job Servers in the set of service providers available to process requests for this service.
   The Administrator completes this action during the next service provider update.

Related Information

Updating service providers [page 1913]

4.3.6.2.2.7 Restarting a service provider

1. Select *Real-Time* **<Access Server>** *Real-Time Services* .
2. Select the service that you want to restart the service provider for. The *Service Provider Status* page opens.

   > Note
   >
   > Select Restart only if the service providers controlled by this Job Server are currently enabled. To verify the status, select the Real-Time Services Configuration tab and view service provider status in the Job Servers for Service section.

3. Click *Restart*.

   The Administrator completes this action during the next service provider update. The Administrator shuts down any service providers controlled by the selected Job Servers and immediately restarts the minimum number of service providers. For example, you might restart a service provider after a computer running its Job Server reboots following a crash.

4.3.6.2.3 Updating service providers

At a specified provider update interval, the Access Server updates service providers. When updating service providers, the Access Server balances the work load—starting or stopping service providers as necessary—and implements other events that you initiated since the last update.

When balancing the work load, the Access Server checks the number of requests in a service queue and the minimum idle time for a service. If the number of requests in a service queue is greater than the number of service providers started, the Access Server tries to start a new service provider. Conversely, if the minimum idle time for a service is more than 10 minutes, the Access Server will shut down a service provider. However, the number of service providers cannot exceed the maximum number of instances configured nor can it be less than the minimum number of instances configured.

When implementing events that you initiated, the Access Server does the following:

- Enables service providers
- Disables service providers
- Reconfigures service providers
- Restarts service providers
- Adds service providers
- Removes service providers

Related Information

Setting the service provider update interval [page 1910]

4.3.6.2.4 Monitoring services

Use the Administrator to monitor services. With the Administrator you can do the following:

- View service status—From the Access Server Status page or the Real-Time Services Status page, view whether a service is running or not. Based on this information, you can begin troubleshooting problems.
- View service provider status—From the Real-Time Services Status page, click a service name to view:
    - The statistics for a particular service.
    - Detailed statistics about each service provider. Using this information, you can monitor and evaluate system performance.
    - The status of all service providers in that service.
- View logs—The Access Server node provides access to current and historical service provider trace and error logs.

Related Information

Viewing the status of services [page 1914]
Service statistics [page 1926]
Service provider statistics [page 1927]
Viewing the statistics for a service provider [page 1915]
Viewing the logs for a service provider [page 1958]

4.3.6.2.4.1 Viewing the status of services

1. Select ▶ *Real-Time* ▸ **`<Access Server>`** ▸ *Real-Time Services* ◀.

   The Administrator opens the Real-Time Services Status page. For each service, this page shows the overall status and statistics about the number of available service providers and the number of started service providers.
2. Verify that the services are working.

Indicator	Description
✓	A green icon indicates that the service is operating properly.
⚠	A yellow icon indicates that some aspect of the service is not working, and that the Access Server is attempting to reestablish the service using error handling.
✖	A red icon indicates that one or more aspects of the service is not working, and the Access Server cannot reestablish the service.
3. If a service shows a yellow or red status, click the service name to see more information.

Related Information

Service statistics [page 1926]
Troubleshooting the Administrator [page 1956]

4.3.6.2.4.2 Viewing the statistics for a service provider

1. Select *Real-Time* > `<Access Server>` > *Real-Time Services*.
2. Click the name of the service.

   This page shows the overall statistics for the service, the service providers for the service (listed by Job Server), and the status of each service provider. Start a service to see its service provider status information.
3. Under *Service Provider Status Information*, click the *Process ID* of a service provider to view its statistics.

   The Administrator opens the Service Provider Status page.

   Under Service Provider Status Information, the page shows the statistics for this service provider.

Related Information

Service provider statistics [page 1927]
Viewing the logs for a service provider [page 1958]

4.3.6.3 Creating and monitoring client interfaces

A client is an external application that communicates with SAP Data Services through the Access Server.

There are two types of client interfaces in the Administrator:

- Remote Function Call (RFC) clients
- Message broker clients

Configure RFC clients in the Administrator for real-time jobs that use SAP IDocs. To support these jobs, create an RFC client interface and attach IDoc configuration information to it.

Data Services creates message broker client interfaces when communication occurs between the Access Server and an external application that uses Data Services message client libraries. To monitor message statistics, view the message broker clients of each Access Server as needed.

This section describes configuration and monitoring for each type of client.

For more information about using the Message Client library, see the *SAP Data Services Integrator Guide*.

4.3.6.3.1 RFC clients

You can configure IDoc message sources in the Administrator as well as in the Designer. You can configure other IDoc sources and targets using the Designer.

> i Note
>
> Using the Administrator, create a service for your real-time job that contains an IDoc as a message source before you configure an RFC client.

An RFC client uses the SAP RFC protocol to communicate with the Access Server. An RFC client requires connection information so that an Access Server can register to receive IDocs from an SAP application server. An RFC client can process one or more IDoc types. An RFC client specifies which service will process a particular IDoc type and whether or not the RFC client connection can process an IDoc type in parallel.

The process of creating an RFC client interface for IDocs has two parts:

- Adding an RFC client
- Adding IDoc configurations to an existing RFC client

Configure one RFC client per Access Server. This means that you can process IDocs from one instance of SAP. To process IDocs from more than one instance, configure more than one Access Server.

> i Note
>
> SAP application function modules are responsible for IDoc processing. In SAP Data Services, the RFC client might fail if multiple IDocs are sent from SAP and you previously set the SAP packet size to 1. Therefore:

- Do not enable the option of immediate IDoc dispatch in SAP unless the volume of produced IDocs is very low (no more than one IDoc per minute).
- For batch processing of IDocs, the packet size should never be smaller than 5 or larger than 1000. The following table provides estimates for this parameter:

IDoc Processing Volume	IDocs per day	Packet size
Light	1 to 300	5
Medium	301 to 1000	20
Heavy	1001 to 5000	80
Very Heavy	more than 5000	800

For more information, see the *SAP Data Services Supplement for SAP*.

4.3.6.3.1.1 Adding an RFC client

1. Select *Real-time* › `<Access Server>` › *Client Interfaces*.
2. Click the *Client Interface Configuration* tab.
3. Click *Add*.

The Administrator opens the RFC Client Configuration page.

4. Enter the client configuration information.

Field	Description
RFC program ID	The registered server program ID from transaction SM59.
User name	The user name through which SAP Data Services connects to this SAP application server.
Password	The password for the user account through which SAP Data Services connects to this SAP application server.
SAP application server name	The domain name or IP address of the computer where the SAP application server is running.
Client number	The SAP application client number.
System number	The SAP application system number.
SAP gateway host name	The domain name or IP address of the computer where the SAP RFC gateway is located.
SAP gateway service name	The TCP/IP service name for the SAP application server gateway. Typically, this value is SAPGW and the system number. It can also be a service number, for example 3301.
Use sapnwrfc.ini	Select to use an sapnwrfc.ini file, which overrides the datastore settings associated with this RFC client. The default location to place the sapnwrfc.ini file is in the current directory of the process being executed (%LINK_DIR/bin).
Destination name	If using an sapnwrfc.ini file, enter the destination name to reference.

5. Click *Apply*.

For more information, see the *SAP Data Services Supplement for SAP*.

Related Information

Supplement for SAP: Using the sapnwrfc.ini file [page 2439]

4.3.6.3.2 Adding IDoc configurations to an RFC client

After you create an RFC client, you can list the IDoc types that you want to receive.

4.3.6.3.2.1 Adding an IDoc configuration to an RFC client

1. Select *Real-time* `<Access Server>` *Client Interfaces* .

2. Click the *Client Interface Configuration* tab.
3. Click the name of an existing RFC client interface.

   The RFC Client Configuration page opens.
4. Click the *Supported IDocs* link.
5. Click *Add*.
6. Enter IDoc information:
   a) In the *IDoc Type* box, enter the IDoc type that this SAP application server will send to the Access Server.
   b) In the *Service Name* box, enter the name of the service that will process this IDoc.

      The service identifies the job that processes this IDoc.
   c) If you want the Access Server to read IDocs (of this type and from the specified SAP source) in parallel, check the *Parallel Processing* check box.

      Real-time services that contain an IDoc message source can be processed one at a time or in parallel. The *Parallel Processing* option allows you to increase the number of IDoc source messages processed per minute for the IDoc type specified. This option is disabled by default. The option allows the Access Server to send an IDoc to a service queue (where it waits for a service provider) and continue with the next IDoc without waiting for reply. The maximum number of outstanding IDoc requests in the queue is the number of IDocs received or four, whichever is smaller.

      > i Note
      >
      > Where a strict IDoc processing sequence is required, do not use the Parallel Processing option.
7. Click *Apply*.
8. (Optional) Select *Real-time* > **`<Access Server>`** > *Client Interfaces* .
9. From the Client Interface Status page, select the check box next to the new *RFC* client and click *Start*.

   The Administrator starts the RFC client. A green indicator signifies that the client is running. Detailed status information is provided in the Status column.

Related Information

Configuring and monitoring real-time services [page 1907]

4.3.6.3.2.2 Closing connections to an RFC client interface

1. Select *Real-time* > **`<Access Server>`** > *Client Interfaces* .
2. Select the check box next to the RFC client you want to disconnect.

   If you choose *Shut down*, the Access Server allows the clients to finish processing any active requests before closing the connection. The Access Server responds with an error to any new requests that arrive during that interval.

   If you choose Abort, the Access Server closes the connection to the client without responding to requests currently being processed.

3. Click *Shut down* or *Abort*.

4.3.6.3.3 Message Broker clients

A Message Broker client uses an XML message to communicate with an Access Server.

Message Broker clients include:

- External applications
- Adapters
- Service providers

Use the Administrator to monitor Message Broker clients.

4.3.6.3.4 Monitoring client interfaces

From the Client Interface Status page, you can view the overall status of all client connections.

4.3.6.3.4.1 Viewing the overall status of client connections

1. Select ▶ *Real-time* ▶ `<Access Server>` ▶ *Client Interfaces* ◢.
2. Verify that the RFC client connections are working.

Indicator	Description
✓	A green icon indicates that each client of this type has an open connection with Access Server.
⚠	A yellow icon indicates that at least one client of this type is disconnecting.
✗	A red icon indicates that the Access Server could not reserve the specified port to listen for client requests.

If an RFC client interface has a red status:

a) View the Status column and click the name of the client to view statistics about the particular client connection with a problem.

b) If you want to restart, abort, or shut down a client interface, click the Back button in the navigation bar. The Administrator returns to the Client Interface Status page.

c) Click *Start*, *Abort*, or *Shutdown*.

Related Information

Finding problems [page 1956]

4.3.6.3.4.2 Monitoring Message Broker clients

Select *Real-time* › **`<Access Server>`** › *Client Interfaces* .

Under *Message Broker Clients*, this page lists each message broker client that has registered with the Access Server along with statistics for that client.

> **Note**
>
> The first client in this list is the Administrator. You registered with the Access Server when you added connection information to the Administrator.

Message broker client interface information includes:

Item	Description
Name	The name of the client.
Time Connected	The total time that this client has been connected to the Access Server.
Last Message Received	The length of time since the Access Server has received a message from this client.
Last Message Sent	The length of time since the Access Server has sent a message to this client.
Received Messages	The number of messages that the Access Server has received from this client.
Sent Messages	The number of messages that the Access Server has sent to this client.

4.3.7 Real-Time Performance

About this section

This section discusses the Access Server parameters, statistics for services and service providers, and how to tune the performance of services and service providers.

Related Information

Configuring Access Server output [page 1921]
Service configuration parameters [page 1923]
Service statistics [page 1926]
Service provider statistics [page 1927]
Using statistics and service parameters [page 1928]

4.3.7.1 Configuring Access Server output

You can configure the Access Server to control its operation and output such as sending specific event information to its trace log.

SAP Data Services installation includes a server configuration utility called the Server Manager. The Server Manager allows you to view and change the following Access Server information:

Option	Description
Directory	The location of the configuration and log files for this instance of the Access Server. Do not change this value after the initial configuration.
Communication Port	The port on this computer the Access Server uses to communicate with the Administrator and through which you can add additional configuration information to an Access Server. Make sure that this port number is not used by another application on this computer.
Parameters	Command-line parameters used by the Data Services Service to start this Access Server. For development, consider including the following parameters: `-P -T16` where `-P` indicates that trace messages are recorded, and `-T16` indicates that the Access Server collects events for services and service providers. These parameters are described in the next table.
Enable Access Server	An option to control the automatic start of the Access Server when the Data Services Service starts.

4.3.7.1.1 Configuring an Access Server

1. Open Data Services Server Manager.
2. Navigate to the Access Server tab and click on *Configuration Editor*.

The *Access Server Configuration Editor* opens.

3. Click *Add* to configure a new Access Server or select an existing Access Server, then click *Edit* to change the configuration for that Access Server.
4. Make the appropriate changes in the *Access Server Properties* window.
5. Click *OK* to return to the *Access Server Configuration Editor*.
6. Click *OK* to return to the Server Manager.
7. Click *Restart* to stop and start the Data Services Service with the new Access Server configuration.

The following parameters are available to control the operation and output of an Access Server:

Parameter	Description	
`-A`	Specifies the communication port for an Access Server. The default value is `-A4000`.	
`-C`	Disables display output.	
`-H`	Prints the parameter list to the console.	
`-P`	Enables trace messages to the console and log.	
`-R` **`<root_directory>`**	Indicates the location of the Access Server directory.	
`-T` **`<value>`**	Determines the type of tracing information displayed in the console and the Access Server log. Use any value or any combination of values:	
	1	system
	2	real-time service flow
	4	client
	8	transaction
	16	service
	64	administration
	128	request
	For example, to enable tracing for both system-level and service-level operations, include the value 17 after the `T` parameter.	
`-V`	Displays the version number of the Access Server.	
`-VC`	Displays communication protocol and version number.	
`-X`	Validates the Access Server configuration without launching the Access Server.	

The `-A` and `-R` parameters can also be set using the Server Manager.

The -P and -T parameters can be set using the Administrator. Select *Real-Time* **<Access Server>** *Logs-Current* , and click the *Access Server Log Configuration* tab.

4.3.7.2 Service configuration parameters

Each service contains configuration parameters that control how the Access Server dispatches requests to the assigned real-time job. These parameters determine how the system handles errors that occur during operation.

Often, requirements during development differ from requirements during production. Therefore, the values of your configuration parameters differ during development and production. To ensure that the system works as expected, test the values before committing the Access Server configuration to production use.

Parameters control different categories of Access Server operation:

- Service startup behavior
- High-traffic behavior
- Response time controls

Related Information

Adding a service [page 1907]
Service startup behavior [page 1923]
High-traffic behavior [page 1924]
Response time controls [page 1925]

4.3.7.2.1 Service startup behavior

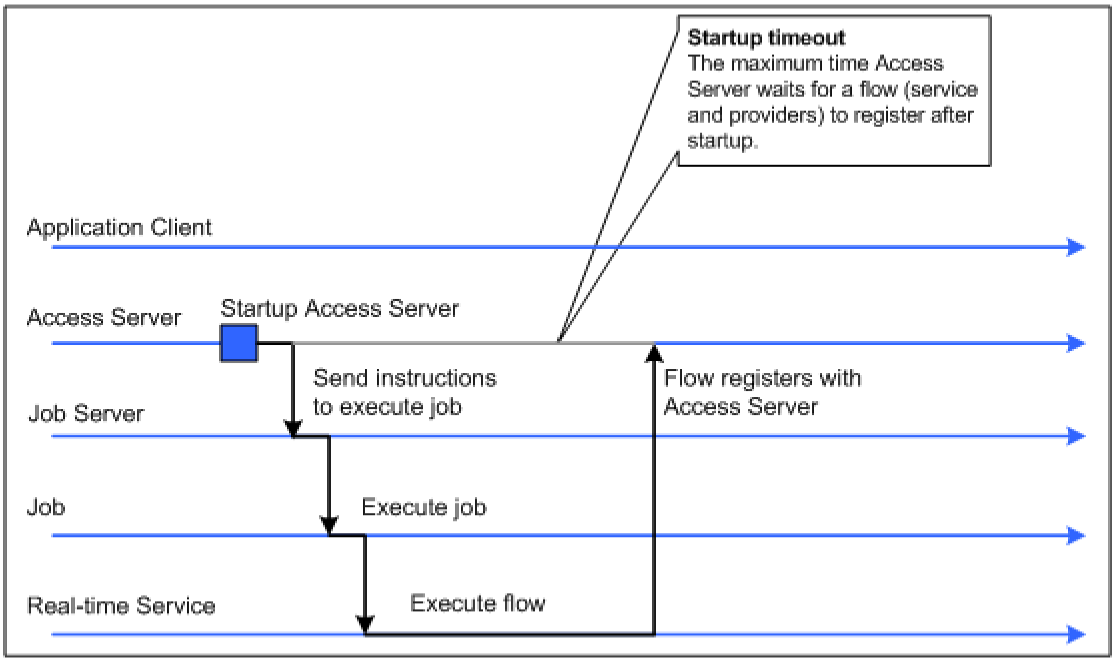
Use two parameters to configure how the Access Server starts service providers associated with a particular service:

- *Startup timeout*—The maximum time that the Access Server waits for a flow (service and its providers) to register after startup.
- *Recycle request count max*—The number of requests that the Access Server sends to a given flow before automatically recycling.

When the Access Server starts, it immediately starts the service providers for each service. If you want the Access Server to start more than one instance of a service to process a particular type of message, you must define more than one service provider for the service.

The Job Servers launch the jobs, which in turn initiate their corresponding real-time services. The first operation of each real-time service is to register with the Access Server.

If an error occurs and a real-time service fails to register, the Access Server instructs the Job Server to restart the job. The Access Server waits the length of time that you configure as the startup timeout before instructing the Job Server to start the job again. The startup timeout is in seconds. The Access Server continues to instruct the Job Server to restart the job until the real-time service registers.


Startup timeout
The maximum time Access Server waits for a flow (service and providers) to register after startup.
Application Client
Access Server
Startup Access Server
Send instructions to execute job
Flow registers with Access Server
Job Server
Job
Execute job
Real-time Service
Execute flow

You can also control how many requests a particular service provider processes. After a provider processes the number of requests specified by the Recycle request count max parameter, the Access Server automatically recycles the service provider—that is, the Access Server automatically stops the current instance of the real-time service and starts a new instance of that service. Setting this parameter to a higher value increases the time that the service provider is available to accept requests for processing. Setting this parameter to a lower value refreshes any data cached in the real-time service more often.

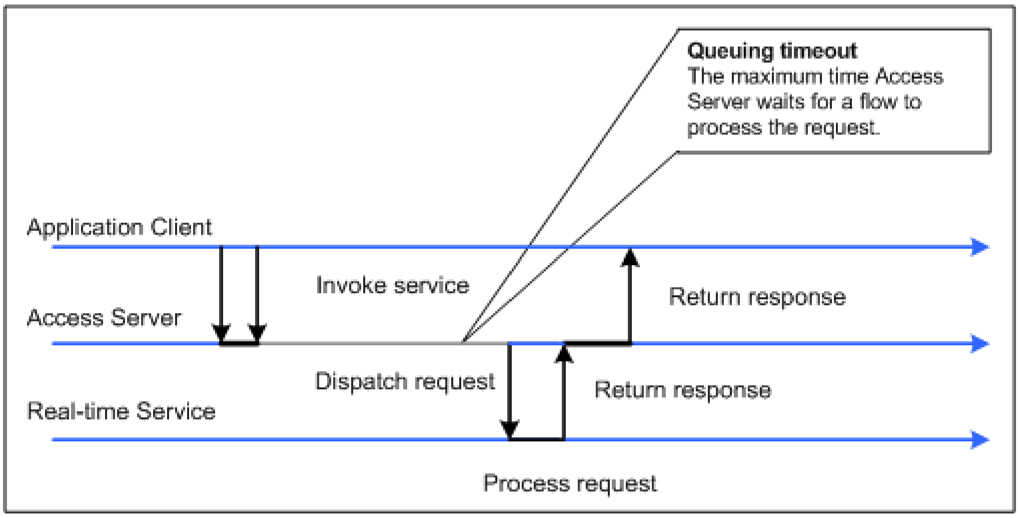
4.3.7.2.2 High-traffic behavior

Use the Queuing Timeout parameter to specify the maximum amount of time the client application must wait for a request to be processed.

If the number of requests the Access Server receives for a particular service exceeds the number of registered service providers that can process those requests, the Access Server queues the requests in the order they are received. When a service provider completes processing a request and responds to the Access Server, the Access Server dispatches the next request in the queue for that service to the open service provider.

If there are many requests and the queue causes requests to exceed the queuing timeout, the Access Server removes the oldest request from the queue and responds to the client with an error indicating that the request failed. You can use the queuing timeout to ensure that the client receives a timely response, even during high-traffic periods.

The queuing timeout is in seconds.



A service experiences high traffic when the available resources cannot process the received requests efficiently. High traffic occurs when the time messages wait to be processed exceeds the time required to process them.

Related Information

Using statistics and service parameters [page 1928]

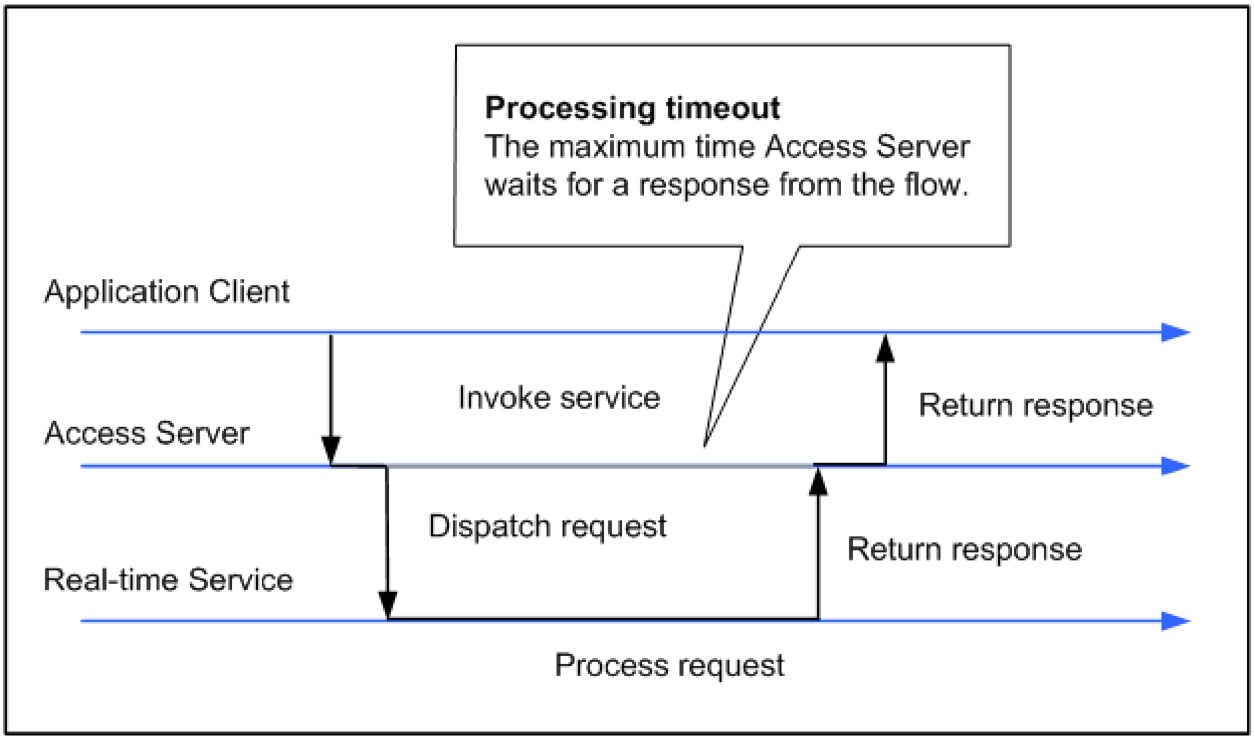
4.3.7.2.3 Response time controls

Use two parameters to configure how long the Access Server waits for responses from service providers for a particular service:

- *Processing timeout*
- *Processing retry count max*

After the Access Server sends a request to a service provider to process, the Access Server waits for the response. If the response does not arrive within the specified processing timeout, the Access Server sends the request to another waiting service provider. The *Processing timeout* is in seconds.

If the first attempt fails, the Access Server will attempt to process the request as many times as you specify in the *Processing retry count max* parameter.



If *Processing retry count max* is set to zero, the maximum response time is equal to the queuing timeout plus the processing timeout.

4.3.7.3 Service statistics

The Real-Time Services Status page for a particular service shows overall statistics.

Statistic	Description
Number of processed requests	The number of requests for this service from any client that the Access Server received and responded to since the last time the Access Server started.
Number of requests in queue	The number of messages that the Access Server has received from a client for this service but has not sent to a service provider for processing. This value reflects the current state of the Access Server.
Maximum queuing time (milliseconds)	The maximum time any request for this service waited after the Access Server received the message and before the Access Server sent the request to a service provider for processing.
Average queuing time (milliseconds)	The average time that requests for this service waited after the Access Server received the request and before the Access Server sent the request to a service provider for processing.

Statistic	Description
Queuing timeout	The number of requests to which the Access Server replied to the client with an error indicating that there was no service provider available to process the request.
Maximum processing time (milliseconds)	The maximum time required to process a request for this service. It is the difference between the time the Access Server sent the request to a service provider and the time that the Access Server responded to the client. The processing time does not include time the request spent in a queue waiting to be sent to a service provider.
Average processing time (milliseconds)	The average time required to process a request for this service. It is the difference between the time the Access Server sent the request to a service provider and the time that the Access Server responded to the client. The processing time does not include time the request spent in a queue waiting to be sent to a service provider.
Processing timeouts	The number of requests that the Access Server sent to a service provider and did not receive a response before exceeding the processing timeout. These requests are either successfully processed by another service provider, or if they are left unprocessed beyond the time indicated by the queuing timeout parameter, the Access Server returns an error to the client.

4.3.7.4 Service provider statistics

The Service Provider Status page shows the statistics for an instance of a real-time service.

When SAP Data Services measures a statistic "from the start," the value does not restart when the Access Server restarts the service provider. The value restarts when the Access Server restarts.

When the software measures a statistic "for the current service provider," the value restarts when the Access Server restarts the service provider, either due to error or when the service provider reaches the maximum number of requests defined for the service.

Statistic	Description
Max processing time (milliseconds)	The longest time it took between when the Access Server sent a message to this service provider and when the service provider returned a response.
Average processing time (milliseconds)	The average time it took between when the Access Server sent a message to this service provider and when the service provider returned a response. If you are running more than one service provider for this service, compare this statistic with the same statistic from the other instances. If this instance is significantly different, look for processing constraints on the computer where this instance runs.

Statistic	Description
Processed requests (for the current service provider)	The number of requests that the Access Server sent to this service provider to which the service provider responded.
Processed requests (since start)	The number of requests that the Access Server sent to this service provider to which the service provider responded.
Error replies received from the start	The number of requests that the Access Server sent to this service provider to which the service provider responded with an error.
Communication errors encountered from the start	The number times that the communication link between the Access Server and this service provider failed.
Timeout errors encountered from the start	The number of times the Access Server sent requests to this service provider and did not receive a response within the time specified by the processing timeout.
Service provider connections (restarts) from the start	The number of times the Access Server restarted this service provider when it did not receive a response from the service provider.
The last time of a successful flow launch	The system time when the Access Server last started the real-time service associated with this service provider. If the Access Server never successfully started an instance of this service provider, the value is "N/A." This time is from the computer running the Access Server.
Time since start attempt	The amount of time since the Access Server last attempted to start this service provider. This value reflects successful and unsuccessful attempts.
Time since last request start	The amount of time since the Access Server last sent a request to this service provider. This value reflects successful and unsuccessful attempts.

4.3.7.5 Using statistics and service parameters

You can use the statistics for a service to tune the service parameters.

Statistic	Description
Average and maximum processing time	If the average or maximum processing time for a service provider is equal or close to the processing timeout value resulting in processing timeouts, consider increasing the processing timeout parameter for the service.
Maximum queuing time	In a tuned system, the maximum and average queuing times should be close together, the difference being an indication of the traffic distribution for this service. Values should not approach the value of the queuing timeout parameter listed for the service.

Statistic	Description
	If the maximum queuing time for a service provider is equal or close to the queuing timeout parameter and there are queuing timeouts listed, consider the following changes: ● Increase the queuing timeout parameter for the service. ● Increase the number of service providers available, either controlled by the same Job Server host or by a different Job Server. If you find that the average time in the queue is longer than the average processing time, the traffic for this service is too high for the resources provided. Consider running multiple service providers to process the same message type. You can add the same job many times in the service list, or you can add the same job controlled by a different Job Server on a separate computer to the service list. If you find that the average queuing time is growing, consider increasing the queuing timeout or adding processing resources.
Processing timeouts	If you see processing timeouts and service providers restarting successfully, consider increasing the number of processing retries allowed for the service.

4.3.8 Profile Server Management

This section describes how to use the Administrator to manage the data in the profiler repository and manage tasks on the profiler server.

The Data Profiler executes on a profiler server to provide the following data profiler information that multiple users can view:

- Column analysis—This information includes minimum value, maximum value, average value, minimum string length, and maximum string length. You can also generate detailed column analysis such as distinct count, distinct percent, median, median string length, pattern count, and pattern percent.
- Relationship analysis—This information identifies data mismatches between any two columns for which you define a relationship, including columns that have an existing primary key and foreign key relationship.

You can execute the Data Profiler on data contained in databases and flat files. Databases include DB2, Oracle, SQL Server, SAP Sybase, and Attunity Connector for mainframe databases. See the *Release Notes* for the complete list of sources that the Data Profiler supports.

This section assumes that you have already installed SAP Data Services, which includes the Data Profiler.

Related Information

Designer Guide: Data Assessment, Using the Data Profiler [page 433]

4.3.8.1 Defining a profiler repository

The Data Profiler repository is a set of tables that holds information about your data that the Data Profiler generates.

1. Create a database to use as your profiler repository.

   The profiler repository can be one of the following database types: DB2, MySQL, Oracle, Microsoft SQL Server, SAP HANA, or SAP Sybase.
2. Create a profiler repository with the Repository Manager.

   Select *Profiler* in the *Repository type* option.
3. Associate the profiler repository with a Job Server with the Server Manager.

Related Information

Administrator Guide: Using the Repository Manager [page 79]
Administrator Guide: Using the Server Manager on Windows [page 84]
Administrator Guide: Using the Server Manager on UNIX systems [page 90]

4.3.8.2 Profiler task parameters

Set configuration parameters to control the amount of resources that profiler tasks use to calculate and generate profiler statistics.

> ℹ Note
>
> If you plan to use Detailed profiling or Relationship profiling, ensure that you use the Server Manager to specify a pageable cache directory that:
>
> - Contains enough disk space for the amount of data you plan to profile.
> - Is on a separate disk or file system from the system where SAP Data Services is installed.

Related Information

Administrator Guide: Using the Server Manager on Windows [page 84]
Administrator Guide: Using the Server Manager on UNIX systems [page 90]

4.3.8.2.1 Configuring profiler task parameters

1. Select *Management* ▸ *Profiler Configuration* to access the Profiler Configuration page.
2. Keep or change the parameters values listed on the Profiler Configuration page.

The Profiler Configuration page groups the parameters into the following categories:

- Task execution parameters
- Task management configuration parameters

Related Information

Task execution parameters [page 1931]

Task management configuration parameters [page 1932]

4.3.8.2.2 Task execution parameters

The Profiler Configuration page groups the Task execution parameters into subcategories Reading Data, Saving Data, and Performance.

Task execution subcategory	Parameter	Default value	Description
Reading Data	Profiling size	All	The maximum number of rows to profile. You might want to specify a maximum number of rows to profile if the tables you profile are very large and you want to reduce memory consumption.
Reading Data	Sampling rows	1	Profiles the first row of the specified number of sampling rows. For example, if you set *Profiling size* to 1000000 and set *Sampling rows* to 100, the Profiler profiles rows number 1, 101, 201, and so forth until 1000000 rows are profiled. Sampling rows throughout the table can give you a more accurate representation rather than profiling just the first 1000000 rows.
Saving Data	Number of distinct values	100	The number of distinct values to save in the profiler repository.
Saving Data	Number of patterns	100	The number of patterns to save in the profiler repository.
Saving Data	Number of days to keep results	90	The number of days to keep the profiler results in the profiler repository.

Task execution subcategory	Parameter	Default value	Description
Saving Data	Number of records to save	100	The number of records to save in the profiler repository for each attribute.
Saving Data	Rows per commit	5000	The number of rows to save before a commit is issued.
Performance	Degree of Parallelism	2	The number of parallel processing threads that the profiler task can use.
Performance	File processing threads	2	The number of file processing threads for file sources.

4.3.8.2.3 Task management configuration parameters

The Profiler Configuration page groups the Task management configuration parameters into subcategories, basic and advanced.

Task management subcategory	Parameter	Default value	Description
Basic	Maximum concurrent tasks	10	The maximum number of profiler tasks to run simultaneously.
Basic	Refresh interval (days)	0	The number of days that must elapse before a profiler task is rerun for the same table or key columns when the user clicks the *Submit* option. The *Submit* option is on the *Submit Column Profile Request* and *Submit Relationship Profile Request* windows in the Designer. The default is 0, which is to always rerun the profiler task when the user clicks the *Submit* option. In other words, there is no limit to the number of Data Profiler tasks that can be run per day. To override this interval, use the *Update* option on the Profile tab of the View Data window in the Designer.
Advanced	Invoke sleep interval (seconds)	5	The number of seconds to sleep before the Data Profiler checks for completion of an invoked task. Invoked tasks run synchronously, and the Data Profiler must check for their completion.
Advanced	Submit sleep interval (seconds)	10	The number of seconds to sleep before the Data Profiler attempts to start pending tasks. Pending tasks have not yet started because the maximum number of concurrent tasks was reached.
Advanced	Inactive interval (minutes)	2	The number of minutes a profiling task can be inactive before the Data Profiler cancels it.

4.3.8.3 Monitoring profiler tasks using the Administrator

You can monitor your profiler task by name in either the Designer or the Administrator.

On the Administrator, you can see the status of profiler tasks, cancel profiler tasks, or delete a profiler task with its generated profile statistics.

1. Expand the *Profiler Repositories* node.
2. Click the profiler repository name.
3. The Profiler Tasks Status window displays.

   This status window contains the following columns:

Column	Description
Select	If you want to cancel a profiler task that is currently running, select this check box and click *Cancel*. If you want to delete a profiler task and its profiler data from the profiler repository, select this check box and click *Delete*. If you click *Delete* on a running task, the Profiler cancels the task before it deletes the data.
Status	The status of a profiler task can be: ○ Done—The task completed successfully. ○ Pending—The task is on the wait queue because the maximum number of concurrent tasks has been reached or another task is profiling the same table. ○ Running—The task is currently executing. ○ Error—The task terminated with an error.
Task Name	The name of the profiler task. The name is a link to the Profiler Task Items report.
Description	The names of the tables on which the profiler task was run.
Run #	The identification number for this profiler task instance.
Last Update	The date and time that this profiler task last performed an action.
Status Message	Is blank if the profiler task completed successfully. Displays an error message if the profiler task failed.

4. Click the task name to display the Profiler Task Items report, which displays the profiling type that was done for each column.

   This Profiler Task Items report contains the following columns:

Column	Description
Status	The status for each column on which the profiler task executed. The status can be: ○ Done—The task completed successfully.

Column	Description
	○ Pending—The task is on the wait queue because the maximum number of concurrent tasks has been reached or another task is profiling the same table. ○ Running—The task is currently executing. ○ Error—The task terminated with an error.
Item	The column number in the data source on which this profiler task executed.
Job Server	The machine name and port number of the Job Server where the profiler task executed.
Process ID	The process ID that executed the profiler task.
Profiling Type	Indicates what type of profiling was done on each column. The Profiling Type can be: ○ Single Table Basic—Column profile with default profile statistics. ○ Single Table Detailed—Column profile with detailed profile statistics. ○ Relational Basic—Relational profile with only key column data. ○ Relational Detailed—Relational profile with data saved from all columns.
Datastore	The name of the datastore.
Source	The name of the data source (table, flat file, or XML file).
Column	The name of the column on which the profiler task executed.
Last Update	The date and time that this profiler task last performed an action.
Status Message	Is blank if the profiler task completed successfully. Displays an error message if the profiler task failed.

Related Information

Designer Guide: Data Assessment, Monitoring profiler tasks using the Designer [page 442]

4.3.9 RFC Server Management

Data Services uses the SAP RFC Server Interface for the following tasks:

- Scheduling SAP jobs
- Reading from SAP Open Hub destinations
- Loading SAP NetWeaver BW
- Viewing Data Services logs from SAP NetWeaver BW

This section describes how to use the RFC Server Interface in the Administrator.

To access the RFC Server Interface, in the Administrator expand the *SAP Connections* node and click *RFC Server Interface*. Two tabs display: *RFC Server Interface Status* and *RFC Server Interface Configuration*.

4.3.9.1 64-bit platform prerequisites

To use the RFC Server Interface on 64-bit UNIX platforms, first confirm that your environment is configured correctly:

1. Ensure that the 64-bit Java Development Kit (JDK) is installed.
2. Ensure that SAP Data Services is correctly installed with a supported web application server.
3. Export the JAVA_HOME environment variable, pointing to the location of the 64-bit JDK.
4. Restart your web application server.

For the latest update of these configuration steps, see *SAP Note 1394367*.

4.3.9.2 Adding an RFC server interface

1. Select *SAP Connections* › *RFC Server Interface*.
2. Click the *RFC Server Interface Configuration* tab.
3. Click *Add*.
   The Administrator opens the RFC Server Configuration page.
4. Enter the configuration information.

   All options except the *RFC Program ID*, *SAP Gateway Host Name*, and *SAP Gateway Service Name* must match the SAP Applications datastore settings.

Field	Description
RFC program ID	The registered server program ID in the SAP RFC destination to which this RFC Server will connect.
User name	The user name through which SAP Data Services connects to this SAP application server. Use the same user name used to create the SAP BW Source datastore.
Password	The password for the user account through which SAP Data Services connects to this SAP application server.
SAP application server name	The domain name or IP address of the computer where the SAP application server is running.
Client number	The SAP application client number.
System number	The SAP application system number.
SAP gateway host name	The domain name or IP address of the computer where the SAP RFC gateway is located.
SAP gateway service name	The TCP/IP service name or service number for the SAP application server gateway. Typically, this value is SAPGW plus the system number.

Field	Description
SAP gateway connection count	The number of TCP/IP connections to the SAP gateway host.

5. Click *Apply*.

The Administrator adds the RFC server interface definition and returns to the *RFC Server Interface Status* page.

After adding the interface, you must manually start it from the *RFC Server Interface Status* page.

4.3.9.3 Starting or stopping an RFC server interface connection

After adding an interface, you must start it manually. To start the interface connection, on the *RFC Server Interface Status* page, select the check box for the interface to start and click *Start*.

To stop an interface connection:

1. Select the check box for the RFC server to disconnect.
2. Click *Abort* or *Shut down*.
   - If you choose *Abort*, the RFC server closes the connection to SAP BW without responding to requests currently being processed.
   - If you choose *Shut down*, the RFC server finishes processing any active requests before closing the connection. Any new requests to this RFC server will result in an error.

4.3.9.4 Monitoring RFC server interfaces

From the RFC Server Interface Status page (expand *SAP Connections* and click *RFC Server Interface*), you can access the status, statistics, and logs for all interface connections.

4.3.9.4.1 Viewing the status of interface connections

The following tables describes the status icons.

Indicator	Description
✅	A green icon indicates that each client of this type has an open connection with SAP BW server.
⚠	A yellow icon indicates that at least one client of this type is disconnecting.
❌	A red icon indicates that there was a problem starting the RFC server or processing an SAP BW request.

If an RFC server interface has a red status, view the *Status* column and click the name of the interface to view the log details.

To restart, abort, or shut down an interface, click the Back button in the navigation bar to return to the RFC Server Interface Status page.

Related Information

Finding problems [page 1956]

4.3.9.4.2 Monitoring interface statistics

The *RFC Server Interface Statistics* section lists each configured interface and statistics for that interface:

Item	Description
Name	The name of the RFC server interface.
Time connected	The total time that this interface has been connected to the SAP BW server.
Last message received	The length of time since the RFC server has received a message from SAP BW.
Last message sent	The length of time since the RFC server has sent a message to SAP BW.
Received messages	The number of messages that the RFC server has received from SAP BW.
Sent messages	The number of messages that the RFC server has sent to SAP BW.

4.3.9.4.3 Viewing RFC server interface logs

You can view the last three logs for an interface, each of which is 500 Kb in size. To view the logs for an interface, from the *RFC Server Interface Status* page, click the name of the interface. A page with two tabs displays:

- *RFC Server Log Viewer*: This tab displays the most recent log for the interface with the log path displayed at the top of the window.
- *RFC Server History Log*: This tab displays the previous two logs for the interface. Click a log name to view it.

4.3.9.5 Removing one or more RFC server interfaces

1. Under the *SAP Connections* node, click *RFC server Interface*.
2. On the *RFC Server Interface Configuration* tab, select the check box for one or more interfaces.
3. Click *Remove*.

4.3.10 Adapters

About this section

This section describes how to add an adapter to the SAP Data Services system, how to start an adapter instance, and how to monitor an adapter's operation instances.

Related Information

Overview of adapters [page 1938]
Adding and configuring adapter instances [page 1939]
Starting and stopping adapter instances [page 1943]
Monitoring adapter instances [page 1944]

4.3.10.1 Overview of adapters

An SAP Data Services adapter is a Java program that allows Data Services to communicate with front-office and back-office applications. Depending on the adapter implementation, adapter capabilities include the ability to:

- Browse application metadata.
- Import application metadata into the repository.
- Move batch and real-time data between Data Services and information resource applications.

Adapters can handle the following types of metadata: tables, documents, functions, outbound messages, and message functions. Each of these can be used in real-time or batch jobs. Outbound messages and message functions are the only objects that include operations.

An adapter can process several predefined operations. An operation is a unit of work or set of tasks that the adapter completes. Operations include:

- Taking messages from an application and send them to a real-time service for processing, possibly returning a response to the application.
- Taking messages from a real-time service and send them to an application for processing, possibly returning a response to the real-time service.
- Taking messages produced by a function call inside a real-time service, send the messages to an application, and return responses to the function.

An adapter connects Data Services to a specific information resource application. You can create one or more instances of an adapter. Each adapter instance requires a configuration file. That configuration file defines the operations available.

All adapters communicate with Data Services through a designated Job Server. You must first install an adapter on the Job Server's computer before you can use the Administrator and Designer to integrate the adapter with Data Services. See your specific adapter's documentation for its installation instructions.

After installing the adapter, configure its instances and operations in the Administrator before creating adapter datastores in the Designer, because you must select an adapter instance name as part of an adapter datastore configuration. It might help to think of the *Adapter Instances* node of the Administrator as part of your adapter datastore configuration.

To enable an adapter datastore connection:

1. Use the Server Manager to configure a Job Server that supports adapters.
2. Use the Administrator to add, configure, and start an adapter instance using the *Adapter Instances* node.
3. Use the Designer to create an adapter datastore and import metadata. Use the metadata accessed through the adapter to create batch and/or real-time jobs.

Related Information

Administrator Guide: Using the Server Manager on Windows [page 84]
Designer Guide: Datastores, Adapter datastores [page 233]

4.3.10.2 Adding and configuring adapter instances

Use the Administrator to add adapter instance configuration information to SAP Data Services and to edit an existing configuration.

Until you add an adapter interface using the Administrator, you cannot run jobs using information from that adapter.

4.3.10.2.1 Adding an adapter instance

1. Select *Adapter Instances* > `<Job Server>`.
2. Click the *Adapter Configuration* tab.
3. Click *Add*.
4. Click an adapter from the list of those installed on the Job Server with which you are working.

   > Note
   >
   > The HTTP adapter and the Web Services adapter automatically install with every Job Server. Both adapters allow you to call external applications from SAP Data Services, one using the HTTP or HTTPS protocol and the other using the SOAP protocol. Use the Web Services adapter to create outbound calls because it automatically configures and starts when a Job Server is enabled for use with adapters. However, if you want to use the HTTP adapter, you can, but you must build it like any other SAP Data Services adapter.

   For more information about the HTTP adapter see the *HTTP Adapter Guide.*
5. Enter the required information to create an adapter instance.
6. Click *Apply*.

   The Administrator adds the adapter instance to the list available to SAP Data Services.

Related Information

Support for Web Services [page 1945]
Adapter instance configuration information [page 1940]

4.3.10.2.2 Editing an adapter's configuration

1. Select ▶ *Adapter Instances* ▶ `<Job Server>` ◀.
2. Click the *Adapter Configuration* tab.
3. Click the name of the adapter instance that you want to edit.

   The Administrator displays the current configuration information.
4. Edit the configuration information.
5. Click *Apply*.

   The Administrator updates the information.

Related Information

Adapter instance configuration information [page 1940]

4.3.10.2.3 Adapter instance configuration information

Complete the following fields in the Administrator to set up an adapter instance in SAP Data Services.

> **Note**
>
> If your adapter instance is for batch jobs, the Adapter Instance Name is the only required option.

Option	Description
Adapter Instance Name	A unique name that identifies this instance of the adapter. If your adapter instance is for batch jobs, this is the only required option.
Access Server Host	To run an adapter instance in a real-time job, you must configure a service that will be called from a given Access Server. Enter the host ID of the computer running the Access Server for which you will configure a service for a real-time job that contains this adapter instance.
Access Server Port	The communication port of the Access Server host is used to both connect an Access Server to SAP Data Services components and to broker messages with external

Option	Description
	applications. After you log in to the Administrator, select ▸ *Real-Time* ▸ **<Access Server>** ▸ *Client Interfaces* ◂ to view an Access Server's message broker port information.
Adapter Retry Count	The number of times to retry an adapter instance if it fails or crashes. Enter 0 to indicate no retries. Enter a negative number to retry the instance indefinitely.
Adapter Retry Interval	The number of milliseconds to wait between adapter retry attempts.

Classpath	All adapter Java programs require specific jar files in the CLASSPATH to use when starting the javaw.exe. For example:
	• `<LINK_DIR>\lib\acta_adapter_sdk.jar`
	• `<LINK_DIR>\lib\acta_broker_client.jar`
	• `<LINK_DIR>\lib\acta_tool.jar`
	• `<LINK_DIR>\ext\lib\xerces.jar`
	Your adapter program might require different jar files.
	You can change the system CLASSPATH environmental variable, or you can use this option to enter a CLASSPATH parameter for the required jar files. If you use this option, enter, for example:
	`C:\Data Services\lib`
	`\acta_adapter_sdk.jar; Data Services`
	`\lib\acta_broker_client.jar;`
	`    Data Services\lib\acta_tool.jar;`
	`    Data Services\ext\lib\xerces.jar`

Autostart	To enable the adapter interface to start automatically when SAP Data Services starts, set this option to True.
Trace mode	A flag that controls the amount of trace messages that the adapter writes. There are two settings: • *True*—The adapter interface writes additional information messages to help debug problems. • *False*—The adapter interface writes only minimal information messages. The adapter writes trace messages to the **`<adapter_instance_name>`**`_trace.txt` file in the **`<DS_COMMON_DIR>`**`\adapters\logs` directory.
Additional java launcher options	In addition to the classpath, you can use additional options when launching java processes (javaw.exe for Windows and java.exe for UNIX platforms). Here are some examples: • If you do not define a value in this box, the default options for memory usage are: `-Xms128m -Xmx256m.`

Option	Description
	- If you get an out-of-memory error from an adapter, then you can re-configure its instance by editing the additional java launcher options. For example: –`Xms512m -Xmx1024m` - If an adapter requires that you define a system property, do so by editing the additional java launcher options: –`Xms128m -Xmx256m foo="string"`
Adapter type name	(Read-only) The name of the adapter used to create this instance.
Adapter version	(Read-only) The version of the adapter used to create this instance.

Adapter Class	(Read-only) A name that identifies the adapter class. The name depends on the type of adapter:
	- For prepackaged adapters, see the adapter documentation.
	- For custom adapters, this is the adapter's fully qualified Java class name:
	`package_name.class_name`
	where:
	○ **`<package_name>`** is the Java package name for the adapter as defined in the adapter's Java file.
	○ **`<class_name>`** is the Java class name for the adapter as defined in the adapter's Java file.
	For example, suppose the adapter's Java file contains these lines:
	`package com.acta.adapter.SiebelAdapter`
	`public class SiebelAdapter implements`
	Then, the adapter class name is:
	`com.acta.adapter.SiebelAdapter.SiebelAd`
	`apter`

Root Directory	Examine the adapter's root directory name. Edit this name as needed.

4.3.10.2.3.1 Adding operation instances to an adapter instance

1. Select ▸ *Adapter Instances* ▸ **`<Job Server>`** ▸.
2. Click the *Adapter Configuration* tab.

3. Click *Operations* under Dependent Objects. The Adapter Operation Configuration page opens.
4. Click *Add* to configure a new operation.

   Here you can also click the link of an existing operation instance to edit its configuration.
5. Select an operation type from the list.

   The options that appear on this page depend on the adapter's specific design.
6. Click *Apply*.
7. Complete the information on the Adapter Operation Configuration page.

   The options and descriptions that appear on this page depend on the adapter's specific design. Consult your adapter-specific documentation for details.
8. Click *Apply*.

4.3.10.3 Starting and stopping adapter instances

Use the Administrator to start and stop an adapter instance and its operations.

- After you configure an adapter instance, each time you stop and start the Access Server, you stop and start the adapter instance and its operations.
- After you restart an adapter instance, the service that uses it fails to process the next message it receives. Therefore, when you restart an adapter instance, also restart its associated services.

4.3.10.3.1 Starting an adapter instance

1. Select *Adapter Instances* `<Job Server>`.
2. Select the check box next to the adapter instance you want to start.
3. Click *Start*.

   The Administrator starts the adapter instance and all of its operations.

4.3.10.3.2 Stopping an adapter instance

1. Select *Adapter Instances* `<Job Server>`.
2. Select the check box next to the adapter instance you want to stop.
3. Click either *Shutdown* or *Abort*:
   - Select *Shutdown* to stop an adapter and all of its operations gracefully. The adapter will complete any pending operations before shutting down.
   - Select *Abort* if you want to stop all operations immediately. Select *Abort* only if incomplete operations are acceptable.

4.3.10.3.3 Starting or stopping an adapter operation instance

1. Select *Adapter Instances* **`<Job Server>`**.
2. Select the check box next to the operation instance you want to start or stop.

   When you start an adapter instance, its operations will also start. However, you can also start and stop individual operation instances manually using this page.
3. Click either *Start* or *Shutdown*.

4.3.10.4 Monitoring adapter instances

Use the Administrator to monitor adapters and their operations.

4.3.10.4.1 Monitoring the adapter instances and operations

1. Select *Adapter Instances* **`<Job Server>`**.

   The Adapter Instance Status page lists each adapter instance and its operations.
2. Find the overall status of a particular adapter instance or operation by examining the indicators.

Indicator	Description
✔	A green icon indicates that the adapter instance or operation has started and is currently running.
⚠	A yellow icon indicates that the adapter instance or operation is not currently running.
✖	A red icon indicates that the adapter instance or operation has experienced an error.

   For each operation, this page lists four statistics:

Statistic	Description
Requests Processed	The number of requests for this operation instance that were processed. Processing of these requests is complete.
Requests Pending	The number of requests for this operation instance that are still pending. Processing of these requests is not complete.
Requests Failed	The number of requests for this operation instance that have failed. The operation has stopped processing these requests.
Status	For operations, displays error text. You can also find more detailed adapter instance information in the Status column. Possible values include: ○ Initialized

Statistic	Description
	○ Starting ○ Started ○ Shutting Down ○ Shutdown ○ Error text—Displays the last error message that occurred as the adapter instance shut down or indicates that the configuration has changed. To allow the adapter instance to use the changes, restart the adapter instance.

For more detailed information about the adapter instance, view the error and trace log files.

4.3.10.4.2 Monitoring adapter instance statistics

1. Select *Adapter Instances* › `<Job Server>`.
2. Click the name of an adapter instance.

   The statistics for the instance appear. The options and descriptions that appear on this page depend on the adapter's specific design. Consult your adapter-specific documentation for details.

4.3.11 Support for Web Services

For information about using SAP Data Services as both a Web services server and client, see the *Integrator Guide*.

4.3.12 Support for HTTP

About this section

The HTTP functionality is installed with every Job Server. This section describes how to configure and use this functionality with SAP Data Services.

Related Information

Overview [page 1946]

Adapter installation and configuration [page 1946]

4.3.12.1 Overview

Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990 and its use has increased steadily over the years, mainly because it has proven useful as a generic middleware protocol.

HTTP is a request/response protocol. A client sends a request to a server specifying a "request method", a Universal Resource Identifier (URL), and protocol version, followed by a message containing client information and usually body content.

The server responds with a status line including the message's protocol version and a success or error code, followed by a message containing server information and usually body content.

HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80 [19], but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such a guarantee can be used.

HTTP can also utilize a Secure Socket Layer (SSL) to implement security at the protocol level. In this manner, data exchange is protected from any unscrupulous elements.

SAP Data Services supports HTTP in the following manner:

- Data transfer can be done using either HTTP or HTTPS (HTTP with SSL) protocols
- The transport mechanism is always TCP/IP
- Both batch and real-time jobs can request data from HTTP-enabled servers, acting as HTTP clients
- Real-time jobs can be executed via HTTP requests, therefore making SAP Data Services act as an HTTP server

4.3.12.2 Adapter installation and configuration

The ability to handle requests to execute real-time jobs as an HTTP server comes pre-configured with SAP Data Services and it is implemented as a servlet deployed inside the Web Server. The ability to call other services as an HTTP client is implemented as an HTTP adapter and it requires further configuration, as explained in subsequent sections.

4.3.12.2.1 URL for HTTP requests

The SAP Data Services server URL format is:

```
http://<host>:<port>/DataServices/servlet/HTTP?ServiceName={GetService}
```

Where:

- **<host>** is the IP address/host name of the Access Server
- **<port>** is the port number of the Access server

These values are the same as in the URL of the Administrator.

4.3.12.2.2 Configuring the HTTP adapter

When you configure the HTTP adapter you must configure one or more instances of the adapter as well as one or more operation instances for each adapter instance.

Adapter operations identify the integration operations available with the configured adapter instance.

Operations provided with the HTTP Adapter include:

- Request/Reply Operation
  This operation is used to execute a remote HTTP service in the Request Reply mode i.e. it makes the request to the remote machine where the HTTP server is running and wait for the reply.
- Request/Acknowledge Operation
  This operation is used to execute a remote HTTP service in the Request Acknowledge mode i.e. it makes the request to the remote machine where the HTTP Adapter server is running and does not wait for the reply; instead, it sends an acknowledgement if the operation is successful.

All adapters communicate with SAP Data Services through a designated Adapter Manager Job Server. Use the Server Manager utility to configure adapter connections with the Adapter Manager Job Server.

Use the Administrator to add an HTTP adapter to the SAP Data Services system and to edit existing adapter configurations. Until you add the adapter in the Administrator, you cannot run jobs using information from that adapter.

Related Information

Administrator Guide: Using the Server Manager on Windows, To configure Job Servers [page 84]

4.3.12.2.2.1 Adding an adapter instance in the Administrator

1. Select ▶ *Adapter Instances* ▸ **`<Job Server>`** ◀.
2. Click the *Adapter Configuration* tab.
3. Click *Add*.
4. Select the HTTP adapter from the list of those available on this Job Server.
5. Enter the required information to create an HTTP adapter instance.
6. Click *Apply*.

   The Administrator adds the adapter instance to the list of those available to the SAP Data Services system.

Related Information

Adapter instance startup configuration [page 1948]

4.3.12.2.2.2 Adapter instance startup configuration

Complete the following fields in the Administrator to set up an HTTP adapter instance in the SAP Data Services system:

Option	Description
Adapter Instance Name	Enter a unique name that identifies this instance of the HTTP Adapter.
Access Server Host	Enter the host ID of the computer running the Access Server that connects to this adapter instance. This information is used by the web application server.
Access Server Port	The Access Server host's message broker port. After you log into the Administrator for this Access Server, select *Configuration* > *Interfaces* to view message broker port information.
Character Set	Used to convert text characters to and from bytes for data.
Metadata Character Set	Used to convert text characters to and from bytes for metadata.
Adapter Retry Count	The number of times for SAP Data Services to retry the adapter instance if it fails or crashes. Enter a negative number to retry indefinitely or enter 0 for no retries.
Adapter Retry Interval	Sets the number of milliseconds between adapter retry attempts.

Classpath	All adapter Java programs require specific jar files in the CLASSPATH to use when starting the `javaw.exe`. For example:
	`LINK_DIR/lib/acta_adapter_sdk.jar`
	`LINK_DIR/lib/acta_broker_client.jar`
	`LINK_DIR/lib/acta_tool.jar`
	`LINK_DIR/ext/lib/xerces.jar`
	`LINK_DIR/lib/acta_http_adapter.jar`
	`LINK_DIR/lib/jcert.jar`
	`LINK_DIR/lib/jnet.jar`
	`LINK_DIR/lib/jsse.jar`

Autostart	When set to *True*, the adapter interface automatically starts when the Administrator starts.
Trace Mode	Set this flag to control the number of trace messages the adapter writes. There are two settings: • *True*—The adapter interface writes additional information messages to help debug problems. • *False*—The adapter interface writes minimal information messages. The adapter writes trace messages to the **`<adapter_instance_name>`**`_trace.txt` file in the **`<DS_COMMON_DIR>`**`/adapters/logs` directory.
Additional Java Launcher Options	Enable to launch the java process that hosts the adapter.

Option	Description
Application command line parameters	Additional command line parameters used for the `javaw.exe` command line and for the adapter itself. See specific adapter documentation for details.
Adapter Type name	(Read-only) The name of the adapter used to create this instance.
Adapter Version	(Read-only) The version of the adapter used to create this instance.
Adapter Class	(Read-only) A name that identifies the adapter class. The name depends on the type of adapter.
Keystore Password	(Optional) The software checks the integrity of the keystore data only if you enter a password. Without a password, the software does not check the integrity of the keystore data. This value is required if using HTTPS protocol to make requests.

4.3.12.2.3 Configuring an operation instance

4.3.12.2.3.1 Adding an operation instance to an adapter instance

1. Select *Adapter Instances* > **`<Job Server>`**.
2. Click the *Adapter Configuration* tab.
3. Under Dependent Objects, click *Operations*.
4. Click *Add* to configure a new operation. Or, you can click the link of an existing operation instance to edit its configuration.
5. Select an operation type from the list (*Request/Reply* or *Request/Acknowledge*), then click *Apply*. The options that appear depend on the operation specific design.
6. Complete the operation instance configuration form.
7. Click *Apply*.

Related Information

Configuring a Request/Reply operation instance [page 1950]

4.3.12.2.3.2 Configuring a Request/Reply operation instance

When creating or editing a Request/Reply operation instance, you must complete the following fields:

Option	Description
Operation Instance	The unique operation instance name. In the Designer, the Request/Reply operation metadata object will be imported with this name.
Thread Count	The number of copies of Request/Reply operations to run in parallel. For parallel (asynchronous) processing of messages coming from a real-time service, use more than one copy. If the sequence of messages is important (synchronous processing), do not use more than one thread. (Multiple copies of real-time services must be supported by multiple copies of Request/Reply.) The default is 1.
Display Name	The operation instance display name. This display name will be visible in the Designer's metadata browsing window.
Description	A description of the operation instance. This description will display in the Designer's metadata browsing window.
Enable	Set to True for the Adapter SDK to start this operation instance when adapter starts; otherwise, set to False.
Target URL	The URL where you want to send the HTTP request from SAP Data Services (the HTTP server address).
Request Method	The HTTP request method used to submit the request. The possible values are POST and GET.
Content-Type	Used to set the content type header of the request. It specifies the nature of the data by giving type and subtype identifiers.
Content-Language	The ISO code for the language in which the request document is written. For example: 'en' means that the language is English in one of its forms.
Content-Encoding	Specifies the encoding mechanism used for sending the request. Current valid entries are x-compress and x-gzip.
Continue If Untrusted	Specifies whether to continue the operation with an untrusted HTTP server. If True, the operation continues; if False, the operation terminates.
Request DTD	The DTD file name that defines the Request XMLmessage used in the operation.
Request XML Root Element	The name of the XML root element in the Request DTD.
Reply DTD	The DTD file name that defines the Reply XML message used in the operation.
Reply XML Root Element	The name of the XML root element in the Reply DTD.

4.3.12.2.3.3 Configuring a Request/Acknowledgement operation instance

When creating or editing a Request/Acknowledgement operation instance, you must complete the following fields:

Option	Description
Operation Instance	The unique operation instance name. In the Designer, the Request/Acknowledge operation metadata object is imported with this name.
Thread Count	The number of copies of Request/Acknowledgement operations to run in parallel. For parallel (asynchronous) processing of messages coming from a real-time service, use more than one copy. If the sequence of messages is important (synchronous processing), do not use more than one thread. (Multiple copies of real-time services must be supported by multiple copies of Request/Acknowledgement) Default is 1.
Display Name	The operation instance display name. This display name will be visible in the Designer's metadata browsing window.
Description	A description of the operation instance. This description will display in the Designer's metadata browsing window.
Enable	Set to True for the Adapter SDK to start this operation instance when the adapter starts; otherwise, set to False.
Target URL	The URL where you want to send the HTTP request.
Request Method	The HTTP request method to be used for submitting the request. The possible values are POST and GET.
Content-Type	Used to set the content type header of the request. It specifies the nature of the data by giving type and subtype identifiers.
Content-Language	The ISO code for the language in which the request document is written. For example: 'en' means that the language is English in one of its forms.
Content-Encoding	Specifies the encoding mechanism used for sending the request. Current valid entries are x-compress and x-gzip.
Continue If Untrusted	Specifies whether to continue the operation instance with an untrusted HTTP server. If True, the operation instance continues; if False, the operation instance terminates.
Request DTD	The DTD file name that defines the Request XML message used in this operation.
Request XML Root Element	The name of the XML root element in the Request DTD.

Restart the HTTP Adapter instance for the configuration changes to take effect.

4.3.12.2.4 Defining the HTTP adapter datastore

You can use the HTTP adapter in a batch or real-time data flow by selecting one of the following objects:

- An Outbound message (for Request/Acknowledge operations)
- A Message Function (for Request/Reply operations)

However, before you select these objects, you must first define an HTTP adapter datastore in the Designer, and then import the operation instances defined for the HTTP adapter instance. A data flow can then pass a message to one of the adapter operation instances defined in the datastore.

To define an adapter datastore, you must:

- Define a datastore object for each adapter instance.
- Define one function or one outbound message for each operation instance to which you want to pass a message.

The following sections summarize the Designer tasks for defining an adapter datastore. For more details, see the *Designer Guide*.

Related Information

Define a datastore object [page 1952]

4.3.12.2.4.1 Define a datastore object

In the Designer object library, you must define a datastore object for each adapter instance.

4.3.12.2.4.1.1 Defining an HTTP adapter datastore

1. Go to the *Datastores* tab in the object library, right-click and select *New* from the menu.

   The *Create New Datastore* editor appears.
2. Name the datastore. It's recommended that you incorporate "HTTP" into the name.
3. For *Datastore type*, select *Adapter*.

   > Note
   >
   > Datastore configuration options change depending on the type of datastore that you are creating.
4. For *Job Server*, select the Job Server configured to handle your HTTP adapter.
5. For *Adapter instance name*, choose the instance name you configured in the Administrator.
6. Click *OK* to save values and create the datastore.

Related Information

Configuring the HTTP adapter [page 1947]

4.3.12.2.4.2 Importing message functions and outbound messages to the datastore

SAP Data Services can pass messages from a data flow to an operation instance. You must import either a function or an outbound message (depending on the type of operation involved) in the Designer datastore library for each operation instance.

HTTP adapter operations contain the following invocation types:

Operation	Invocation Type
Request/Reply	Message Function
Request/Acknowledge	Outbound Message

4.3.12.2.4.2.1 Importing message functions and outbound messages

1. In the Designer, double-click the datastore that is associated with your HTTP Adapter Instance.

   The Adapter Metadata Browser window opens.
2. Right-click the operation instance that you want to import and select *Import* from the menu.

   The operation instance you selected is added to the datastore.

You can use imported message functions and outbound messages in your real-time data flows.

4.3.12.2.5 Configuring SSL with the HTTP adapter

With Secure Socket Layer (SSL), the HTTP adapter can use secure transport over TCP/IP networks.

To connect to an SSL-enabled web application server other than the packaged Tomcat server, you must obtain the keystore and certificate from that HTTP service provider. The following procedure describes how to configure the client.

4.3.12.2.5.1 Configuring the client to access a public SSL-enabled Web server

1. Generate the client keystore.
2. Obtain a signed certificate from the SSL-enabled web server.
3. Import the certificate into the client keystore.

4.3.12.2.5.2 Generating a keystore file

If you need to generate a keystore file for your web application server, you can use the `keytool` tool included with the Java Runtime Environment.

1. Open a command prompt, and navigate to the `bin` folder for your Java installation.
2. At the command prompt, run the `keytool` command:

   ```
   keytool -genkey -alias tomcat -keyalg RSA -keystore <path/to/keystore.file>
   ```

   Replace **`<path/to/keystore.file>`** with the desired filename and location for the generated keystore file. For example, `D:\tomcat.keystore`.

   > Note
   >
   > If you do not specify the **`-keystore`** parameter, the keystore file is generated in the home directory of the user who executes the command.

3. Enter the password for the keystore.
4. Enter the host name parameter value from the Target URL property of the operation instance for the first and last name.
5. Enter information about your organization and city as appropriate for each prompt.

   This information is displayed to users who attempt to access secure pages within your application.
6. Press `Enter` to automatically use the keystore password as the key password for the certificate.

The keystore file containing the certificate that can be used by your web application server is generated in the specified location.

4.3.12.2.5.3 Configuring SSL on the web application server

For SAP Data Services to work with SSL, the web application server must be configured to support SSL connections. The `server.xml` file can be used to configure the packaged Tomcat application server.

> Note
>
> For other web application servers, refer to the product documentation about how to configure SSL support.

1. Open `server.xml` in a text editor. This file is located in the `Tomcat55\conf` directory at the same level as `LINK_DIR`.

2. Locate the commented `connector` element in the XML:

```
<!-- Define a SSL HTTP/1.1 Connector on port 8443 -->
<!--
<Connector port="8443" maxHttpHeaderSize="8192"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" disableUploadTimeout="true"
  acceptCount="100" scheme="https" secure="true"
  clientAuth="false" sslProtocol="TLS" />
 -->
```

3. Remove the comment (`<!-- -->`) tags around the `connector` element.
4. Add the `keystoreFile` and `keystorePath` attributes into the `connector` element.

```
keystoreFile="<path/to/keystore/file>"
keystorePass="<keystore_password>"
```

5. Restart the Tomcat application server.

4.3.12.2.5.4 Importing a certificate into the client keystore

The HTTP Adapter client internally handles the details of certificate authentication by implementing the X509TrustManager interface and using SSLSocketFactory classes from the HttpsURLConnection class.

When a HTTPS request is made to the SSL-enabled web server, the client requests the server's certificate, which may be issued by some standard authority such as VeriSign.

If, when the HTTP client checks its certificate store (located in `%LINK_DIR%\ext\Jre\lib\security`), the client determines that the certificate is trusted, the client receives all of the requested data from web server.

The HTTP client requires a password to query the local keystore for verification. As part of the adapter configuration, enter this password as the value of the `keystorePassword` parameter.

If the HTTP client encounters an untrusted certificate, the client throws the SSLException to its caller and checks the value of the `continueIfUntrusted` parameter. If the parameter is set to *false*, the SSLException displays with an informative message and gets logged in trace files. Further, the client does not receive any data from the server. If the parameter `continueIfUntrusted` is set to *true*, then the software logs the SSLException in error and trace files and the client receives data from the server. The certificate file `untrust.cer` is downloaded in the user's current working directory or under `LINK_DIR/bin` directory.

To import the certificate file to the JDK certificate keystore, use the keytool command:

```
keytool -import -alias <DESCRIPTION> -file untrust.cer  -keystore <Full path of
Cacerts file in the <java.home>/lib/security/ directory> cacerts  -storepass
<password>
```

where `DESCRIPTION` can be any text in double quotes. The `storepass` expects the same password with which you created the keystore in the web server.

You will also be prompted for a keystore password; enter whatever password you want. The keytool command will print the certificate information and ask you to verify it. Enter **`yes`** to complete the import process.

4.3.13 Troubleshooting the Administrator

About this section

The Administrator provides status and error information. Use this information to discover problems with your implementation and to find the source of those problems. This section describes how you can use the Administrator to find and help resolve job processing issues.

Related Information

Reestablishing network connections [page 1956]
Finding problems [page 1956]
Error and trace logs [page 1957]
Resolving connectivity problems [page 1962]
Restarting the Access Server [page 1963]

4.3.13.1 Reestablishing network connections

When you disconnect from your network and re-connect or otherwise change an IP address (Dynamic IPs), the Administrator encounters a database connection error.

To reestablish network connections for your repository, you can do one of two things:

- Rename the repository in the Administrator. This change forces the Administrator to drop and recreate the connection to the database.
- Restart the Administrator.

4.3.13.2 Finding problems

The Administrator uses colored indicators to show the status of the various system components. Generally, the indicators mean the following:

Indicator	Description
✓	A green icon indicates that the object is running properly.
⚠	A yellow icon indicates that some aspect of this object is not working. Either the Access Server is in the process of its error-handling efforts to reestablish an operation, or the Access Server is waiting for a manual intervention. For example, when you first add a service to the Access Server configuration, the service displays a yellow icon until you manually start the service or until you restart the Access Server.

Indicator	Description
❌	A red icon indicates that one or more aspects of this object is not working, and the error handling efforts of Access Server were not able to reestablish the operation of the object.

When you see a yellow or red icon, the system requires manual intervention. You must:

- Determine which object is not operating properly.
- Determine the cause of the problem.
- Fix the problem.
- Restart the affected service providers if necessary.

4.3.13.2.1 Determining which object is not operating properly

1. In the Administrator, click *Home*.

   If there is an error anywhere in the system, you will see a red indicator next to a repository or Access Server name.
2. If you see a red indicator, click a repository or Access Server name.

   The page for that repository or Access Server appears.
3. Look for another red indicator on objects listed at this level.
4. If you can identify lower-level objects that have a red indicator, repeat the previous two steps.

   When you have identified the lowest level object with the error, you are ready to determine the cause of the error.

4.3.13.2.2 Determining the cause of the error

1. Examine the error log for the affected subsystem, such as a batch job, a service provider, or an adapter interface, or for the Access Server itself.

   Use the timestamp on the error entries to determine which error accounts for the problem that you are experiencing.
2. Cross-reference the error to the trace information.

   When you identify the appropriate error message, you can use the timestamp to determine what other events occurred immediately before the error.

   For example, if an error occurred for a specific service provider, you can use the error timestamp in the service provider error log to look up Access Server activities that preceded the error.

4.3.13.3 Error and trace logs

The Administrator provides access to trace and error log files for each service provider, each batch job that ran, and for the Access Server. Use these detailed log files to evaluate and determine the cause of errors.

4.3.13.3.1 Batch job logs

The Batch Jobs Status page provides access to trace, monitor, and error log files for each batch job that ran during a specified period. For information about setting that period, see Setting the status interval.

For information about monitor logs, deleting logs, and the *Ignore Error Status* button see Monitoring jobs.

4.3.13.3.1.1 Viewing a batch job trace log

The trace log file lists the executed steps and the time that execution began. Use the trace log to determine where an execution failed, whether the execution steps occurred in the order you expect, and which parts of the execution were the most time-consuming.

1. Select *Batch Jobs* `<repository>`.
2. Identify the instance of the job execution in which you are interested by the job name, start time, and so on.
3. Under *Job Information* for that instance, click *Trace*.

   The Administrator opens the Job Trace Log page.

4.3.13.3.1.2 Viewing a batch job error log

The error log file shows the name of the object that was executing when an error occurred and the text of the resulting error message. If the job ran against SAP application data, the error log might also include ABAP errors.

Use the error log to determine how an execution failed. If the execution completed without error, the error log is blank.

1. Select *Batch Jobs* `<repository>`.
2. Identify the instance of the job execution in which you are interested by the job name, start time, and so on.
3. Under *Job Information* for that instance, click *Error*.

   The Administrator opens the Job Error Log page.

4.3.13.3.2 Service provider logs

The Service Provider Status page provides access to the error and trace log files for a service provider. These are the log files produced by the Job Server that controls the service provider.

4.3.13.3.2.1 Viewing the logs for a service provider

1. Select *Real-Time* `<Access Server>` *Real-time Services*.

2. Click the name of a service.

   The Administrator opens the Real-time Service Status page. This page shows a list of service providers for the service and overall statistics for the service.
3. Click the name of the service provider process ID in which you are interested.

   The Administrator opens the Service Provider Status page.
4. Click a link to view the desired service provider log.

   To delete these logs, set the log retention period. To filter the list of real-time services on the Real-time Service Status page by date, set the status interval.

Link	Description
Trace Log	Opens the Trace Log page for the current service provider execution. This link appears only if the real-time service is registered with the Access Server.
Error Log	Opens the Error Log page for the current service provider execution. This page lists errors generated by the software, by the source or target DBMS, or the operating system for job execution. If the error log is empty, the job has not encountered errors in message processing. This link appears only if the real-time service is registered with the Access Server.

The computer running the Job Server stores text files containing the batch and service provider trace, error and monitor logs. If you installed SAP Data Services in the default installation location, these files are located in the <**<DS_COMMON_DIR>**>/Logs/**<JobServerName>**/**<RepoName>** folder.

The name of the log file describes the contents of the file:
<type>_**<timestamp>**_**<sequence>**_**<jobname>**.txt, where:

- **<type>** is trace, monitor, or error.
- **<timestamp>** is the system date and time from when the job created the log.
- **<sequence>** is the number of this job related to all jobs run by this Job Server instance.
- **<jobname>** is the name of the job instance.

Batch job trace and error logs are also available on the *Log* tab of the Designer project area. To see the logs for jobs run on a particular Job Server, log in to the repository associated with the Job Server when you start the Designer.

Related Information

Setting the status interval [page 1877]

4.3.13.3.3 Access Server logs

Trace and error logs for each Access Server are available in *Real-Time* > **<Access Server>** > *Logs-Current* and *Real-Time* > **<Access Server>** > *Logs-History*. In addition, these files are located in the Access Server configuration location, which you specify when you configure the Access Server.

> **Note**
>
> For remote troubleshooting, you can also connect to any Access Server through the Administrator.

4.3.13.3.3.1 To view the current day's logs

1. Select ▶ *Real-time* ▶ `<Access Server>` ▶ *Logs-Current* ◀.
2. This page lists the error log file followed by the trace log file. The Administrator shows the last 100,000 bytes of the Access Server error log or trace log for the current date.

   The date of the file is included in the name:
   - `error_MM_DD_YYYY.log`
   - `trace_MM_DD_YYYY.log`
3. To view a file, click the file name. The Administrator shows the last 100,000 bytes of the Access Server error log or trace log for the current date.
   - The error log contains error information that the Access Server generates.
   - The trace log contains a variety of system information. You can control the information the Access Server writes to the trace log.

Related Information

Configuring the trace log file [page 1960]

4.3.13.3.3.2 Viewing the previous day's logs

1. Select ▶ *Real-Time* ▶ `<Access Server>` ▶ *Logs-History* ◀.
2. This page lists error log files followed by trace log files. The date of the file is included in the name:
   - `error_MM_DD_YYYY.log`
   - `trace_MM_DD_YYYY.log`
3. To view a file, click the file name.

4.3.13.3.3.3 Configuring the trace log file

1. Select ▶ *Real-Time* ▶ `<Access Server>` ▶ *Logs-Current* ◀.
2. Click the *Access Server Log Configuration* tab.
3. Under *Log Contents*, the Administrator lists several trace parameters that control the information that the Access Server writes to the trace file.

Name	Description
Admin	Writes a message when an Access Server connection to the Administrator changes.
Flow	Writes a message when an Access Server exchanges information with a real-time service.
Request	Writes a message when an Access Server receives requests.
Security	Writes a message when an Access Server processes authentication information (IP addresses, user name, or password).
Service	Writes a message when an Access Server starts or stops a service.
System	Writes a message when an Access Server initializes, activates, or terminates.

4. Select the check box next to the name if you want the Access Server to write corresponding messages to the trace log file.
5. Under *Log Tracing*, select the *Enabled* check box.
6. Click *Apply*.

   The Administrator changes the Access Server configuration. The Access Server will now write the selected trace messages to the trace log.

> i Note
>
> Until you set the parameters on this page, the Access Server uses the startup parameters to determine trace options. Each time you restart the Access Server, the startup parameters take precedence over parameters set on this page. You can control the content of this log by setting parameters when configuring the Access Server.

Related Information

Restarting the Access Server [page 1963]
Configuring Access Server output [page 1921]

4.3.13.3.3.4 To delete Access Server logs

1. Select *Real-Time* › `<Access Server>` › *Logs-Current* or *Real-Time* › `<Access Server>` › *Logs-History* .
2. Select the check box next to any log file that you want to delete.

   Alternatively, to delete all of the log files, select the Select all check box.
3. Click *Clear* or *Delete*.

   The Administrator clears the file size for current logs and deletes the selected history files from the display and from the Access Server log directory.

4.3.13.3.4 Adapter logs

For more detailed information about an adapter or an adapter's operations, see the adapter's error and trace log files.

4.3.13.3.4.1 Viewing log files for an adapter instance

1. Select *Adapter Instance* **`<Job Server>`**.
2. Find the adapter instance for which you want to view logs and from the *Log Files* column, click the *Error Log* or *Trace Log* link.
3. The corresponding page opens.

   These log files are also found in the **`<DS_COMMON_DIR>`**`\adapters\log` directory. The error log file is named **`<adapter_instance_name>`**`_error.txt` and the trace log file is named **`<adapter_instance_name>`**`_trace.txt`.

4.3.13.4 Resolving connectivity problems

If you have determined that you have connectivity problems among your real-time system components, consider the following possible failures:

- `Application client cannot connect to Access Server`
  For example, an error appears in the logs generated by your application client or in the command prompt when you execute the client test utility that looks like this:
- `Error: unable to get host address`
  If you specified an IP address and received this error, your network might not support static IP address resolution. Try using the computer name instead.
  Match the port number that you specified in the client test utility (or in the Message Client library call) to the Access Server's port number.
  Make sure that the port that you specified is not in use by other applications on the computer where an Access Server is installed.
- `Access Server cannot connect to Job Server`
  If this error occurs, you would see a red indicator for a service provider and an error log for the Access Server.
  Match the host name and port number of the Job Server for the service being called (configured in the Administrator) to the host name and port number that the Job Server is configured to use (as listed in the Server Manager).
  To make sure that the Job Server is running, check the Windows Task Manager for the `Al_jobserver.exe` and `Al_jobservice.exe` processes or open the Designer, log in to the repository that corresponds to the Job Server, and look for the Job Server icon at the bottom of the window.
- `Job Server cannot start real-time service`
  If this error occurs, the status indicator for the related service and its service provider would be red and you would be able to open the error log file from the Service Provider Status page.
  Make sure that the job is properly indicated for the service called in the Real-Time Service Status page.
  Make sure that the real-time jobs are available in the repository associated with the Job Server.

Make sure the repository and the Access Server are connected to the Administrator and that the repository is available.
If you change the password for your repository database, the Job Server will not be able to start real-time services. To fix this problem, re-register your repository in the Administrator and reconfigure the real-time services.

- `Real-time service cannot register with Access Server`
  If this error occurs, you would see:
  - A red indicator for the service provider.
  - An error log in the Logs-Current page (the startup timeout will eventually be triggered).
  - An error log available from the Service Provider Status page.

  Make sure the Access Server host name correctly identifies the computer where the Access Server is running.
- `Access Server cannot connect back to application client`
  If this error occurs, you would see an error log under the Access Server's Logs-Current node.
  Make sure that the host name and port used by the message broker client to communicate with the Access Server is correct.

4.3.13.5 Restarting the Access Server

To restart the Access Server, you can use either of two methods:

- Controlled Restart
  The Access Server responds to new and queued messages with a shutdown error. It waits for service providers to complete processing existing messages, then returns the responses to those clients. Next, the Access Server closes existing client connections (including adapters), stops, and restarts itself. Finally, the Access Server reads the current configuration settings and restarts services, service providers, and adapters. Restarting the Access Server this way requires as much time as it takes to process requests in the system.
- Abort and Restart
  The Access Server responds to new and queued messages with a shutdown error. It shuts down existing service providers and responds to these messages with a shutdown error. Next, the Access Server closes existing client connections (including adapters), stops, and restarts itself. Finally, the Access Server reads the current configuration settings and restarts services, service providers, and adapters.

4.3.13.5.1 Performing a controlled restart of the Access Server

1. Select *Real-Time* > **`<Access Server>`** > *Status*.
2. Under *Life Cycle Management*, click *Controlled Restart*.
3. Click the *Real-Time Services* node to verify that all services started properly.

   The Access Server allows running services to complete and returns incoming and queued messages to the client with a message that the Access Server has shut down. When all services have stopped, the Access Server stops and restarts itself. The Access Server reads the new configuration settings and starts services as indicated.

If all service providers started properly, the Real-Time Service Status page shows a green indicator next to each service name. A red indicator signifies that a component of the service did not start.

4.3.13.5.2 Performing an abort and restart of the Access Server

1. Select *Real-Time* > **`<Access Server>`** > *Status* .
2. Under *Life Cycle Management*, click *Abort and Restart*.
3. Click the *Real-Time Services* node to verify that all services started properly.

4.4 Metadata Reports

This section describes the overall requirements for enabling and viewing metadata reports including software and configuration requirements and logging in to the Management Console.

4.4.1 Requirements

Use the SAP Data Services Administrator to configure repositories for metadata reporting applications to access.

To make metadata reports available for objects, the metadata reporting applications require:

- A web application server
- JDBC drivers to connect to a repository
- Configured repositories

4.4.2 Repository reporting tables and views

The SAP Data Services repository is a database that stores your software components and the built-in design components and their properties. The open architecture of the repository allows for metadata sharing with other enterprise tools.

Within your repository, the software populates a special set of reporting tables with metadata describing the objects in your repository. When you query these tables, you can perform analyses on your applications.

The following table lists the Metadata reporting tables and what they contain:

Name	Contains
AL_ATTR	Attribute information about native objects
AL_AUDIT	Audit information about each data flow execution

Name	Contains
AL_AUDIT_INFO	Information about audit statistics
AL_CMS_BV	Components of a Business View
AL_CMS_BV_FIELDS	Business Fields within Business Elements in a Business View
AL_CMS_REPORTS	Information that uniquely identifies Crystal Reports reports, SAP BusinessObjects Desktop Intelligence documents, or SAP BusinessObjects Web Intelligence documents
AL_CMS_REPORTSUSAGE	Tables, columns, Business Views, or SAP BusinessObjects Universe Builder universes that a Crystal Reports report, SAP BusinessObjects Desktop Intelligence document, or SAP BusinessObjects Web Intelligence document uses
AL_CMS_FOLDER	Folder names in which a Crystal Reports report, SAP BusinessObjects Desktop Intelligence document, or SAP BusinessObjects Web Intelligence document resides
AL_CMS_UNV	Information that uniquely identifies universes
AL_CMS_UNV_OBJ	Universe classes and the child objects, with the source column and table
AL_HISTORY	Execution statistics about jobs and data flows
AL_INDEX	Index information about imported tables
AL_LANG	Information about native (.atl) objects
AL_LANGXMLTEXT	Information about objects represented in XML format
AL_PCOLUMN	Column information about imported table partitions
AL_PKEY	Primary key information about imported tables
AL_QD_VRULE	Validation rule names
AL_QD_VRULE_OFLOW	Rule name if it cannot fit it into AL_QD_VRULE (if there is an overflow)
AL_QD_STATS	Runtime validation rule statistics
AL_QD_ROW_DATA	Sample row data for which the validation rules have failed
AL_QD_COLINFO	All the column information for the failed validation rules
AL_USAGE	All ancestor-descendant relationships between objects
ALVW_COLUMNATTR	Attribute information about imported columns
ALVW_COLUMNINFO	Information about imported column
ALVW_FKREL	Primary-foreign key relationships among imported tables
ALVW_FLOW_STAT	Execution statistics about individual transforms within data flows
ALVW_FUNCINFO	Information about both native functions and functions imported from external systems
ALVW_MAPPING	Mapping and lineage information for target tables
ALVW_PARENT_CHILD	Direct parent-child relationships between objects
AL_SETOPTIONS	Option settings for all objects

Name	Contains
ALVW_TABLEATTR	Attribute information about imported (external) tables
ALVW_TABLEINFO	Information about imported tables

i Note

This is not the complete list because some repository tables and views are for internal use.

The software automatically creates reporting tables for each new or upgraded repository. Except for AL_USAGE, the software automatically updates all reporting tables.

Related Information

Reference Guide: Metadata in Repository Tables and Views [page 1721]

4.4.3 Logging into the Management Console

To access the metadata reporting applications, first log into the Management Console. The first time you log into the Management Console, use the default user name and password (`admin/admin`). It's recommended that you change the defaults thereafter by updating user roles in the Administrator.

1. The first step depends on your operating system.
   - On Windows, choose *Start* › *Programs* › *SAP Data Services* **`<x.x>`** › *Data Services Management Console*.
     If you encounter an error, check to see whether your web application server is installed and running. If you are using the packaged Tomcat application server and it is running but you cannot log in, see the Troubleshooting the Administrator section.
   - On UNIX or Windows, open a browser, enter the following case-sensitive URL, then press Enter:
     `http://<hostname>:28080/DataServices/`
2. Enter the default user name (`admin`) and password (`admin`) and click *Log in*.

   The Management Console home page opens.
3. To launch one of the metadata reporting applications, click its icon (or name).

   If you are logged in to the Designer, you can also access the Management Console home page as follows:
   - From the *Tools* menu, click *Data Services Management Console*.
   - Click the *Data Services Management Console* tool bar icon.

4.4.3.1 Management Console navigation

After you log in to the Management Console and launch one of the applications, the application name appears under the Management Console banner.

The upper-right side of the main window includes the following links:

- *Home*—Returns to the Management Console home page (for example, to select another application).
- *Settings*—Use the Settings control panel to change a variety of options depending on the application.
- *Logout*—Exits the application and the Management Console and return to the login page.
- Help icon—Opens the *Management Console Guide*.

As you navigate around the applications, notice the top of the right-hand pane often displays a "bread crumb" path to indicate where you are in the application. Depending on the page displayed, sometimes you can click on the bread crumbs to navigate to a different part of the application.

The Administrator, Impact and Lineage Analysis, and Auto Documentation applications also use a navigation tree in the left-hand pane.

Management Console sessions time out after 120 minutes (2 hours) of inactivity.

4.5 Impact and Lineage Analysis Reports

The Impact and Lineage Analysis application provides a simple, graphical, and intuitive way to view and navigate through various dependencies between objects.

Impact and lineage analysis allows you to identify which objects will be affected if you change or remove other connected objects.

For example for impact analysis, a typical question might be, “If I drop the source column Region from this table, which targets will be affected?”

For lineage analysis, the question might be, “Where does the data come from that populates the Customer_ID column in this target?”

In addition to the objects in your datastores, impact and lineage analysis allows you to view the connections to other objects including universes, classes and objects, Business Views, Business Elements and Fields, and reports (Crystal Reports, SAP BusinessObjects Desktop Intelligence documents, and SAP BusinessObjects Web Intelligence documents).

Metadata Integrator

Metadata Integrator is required to view BusinessObjects objects in the Impact and Lineage reports. However, Metadata Integrator is no longer a component available with Data Services, beginning with the 4.1 release.

From Data Services 4.1, you cannot directly configure Metadata Integrator to collect from Business Intelligence platform XI 3.x or 4.0. The Data Services XI 3.x or 4.0 Metadata Integrator can continue to work with the Data Services 4.1 repository; however, this will not work with an SAP HANA repository.

If you have a previous version, you can continue to view impact and lineage for objects from a Business Intelligence platform 3.x or 4.0 system after executing the previous version of Metadata Integrator to load metadata to the Data Services 4.1 repository.

If you are migrating from SAP Data Services XI 3.x or 4.0, and have Business Intelligence platform Metadata Integrator configured to collect from a CMS repository version XI 3.x or 4.0, you must retain the Business Intelligence platform 3.x or 4.0 client machine with Data Services XI 3.x or 4.0 Metadata Integrator installation to continue to collect CMS metadata from Business Intelligence platform XI 3.x or 4.0 systems.

You can continue to use the Data Services-specific impact and lineage.

For full functionality, you will need to use the Metadata/CMS integrator found in SAP Information Steward.

> **Note**
>
> You must add permissions for the Business Intelligence platform 4.0 FP3 user to retrieve the Data Services 4.1 repository password. Otherwise, their Metadata Integrator instance cannot get BusinessObjects objects when running on Data Services 4.0. See the *Data Services Administrator Guide* for more information.

4.5.1 Navigation

From the Management Console home page, view impact and lineage information by clicking the *Impact and Lineage Analysis* link.

The Impact and Lineage Analysis page contains two primary panes:

- The left pane contains a hierarchy (tree) of objects. The top of the tree is the default repository. This pane also includes a search tool.
- The right pane displays object content and context based on how you navigate in both panes.

In general, expand the tree in the left pane by clicking plus signs (+) next to object nodes. Select an object in the tree to learn more about it. Details associated with an object appear in the right pane on one or more tabs. Tabs vary depending on the object you select.

The top level of the navigation tree displays the current repository. (You can change this repository in the Settings control panel; for details, see *Impact and Lineage Analysis Settings control panel* [page 1973].)

Objects in a repository include:

- Datastores—Contain tables and columns.
- CMS server(s)—After you configure the Metadata Integrator, this node contains folders as defined in the Central Management Console, Universes, and Business Views.
  The Universe node displays classes, and each class contains objects. Business Views contain Business Elements, and each Business Element has Business Fields. Any object can contain one or more reports.

To narrow the view of objects to display in the navigation tree or to quickly find specific objects in the repository, use the *Objects to analyze* search feature. You can do one or both of the following:

1. Select a category in the *Select an object type* drop-down list:
   - Table and column
   - Universe
   - Class and object

- Business view
    - Element and field
    - Report

   AND/OR
2. Search for a specific object by typing all or part of the object name in the search field. The search field is not case sensitive, spaces are allowed, and you can use the percent symbol (%) as a wildcard.
3. Click the search icon (binoculars).

Metadata Reports highlights with a red border each object that meets your search criteria.

- To repopulate the CMS navigation tree with any objects that might have been changed (added, deleted, renamed, and so on) in the CMS repository after you display an impact and lineage report, log out and log back in to the Management Console to display the changes.
- If you receive an error such as the following:

  ```
  This image is scaled to 50% of the original image. You could avoid scaling the
  image by allocating more memory to the current java process.
  ```

  it means the java process (JVM) does not have enough memory to process the current task. Circumvent this error by allocating more heap memory to the Java process associated with the Web server.

4.5.1.1 To increase the java heap memory in Windows

1. In the Windows Services control panel, stop the Data Services Web Server.
2. In the installation directory, navigate to \ext\WebServer\conf.
3. Using a text editor, open the wrapper.properties file.
4. To allocate a minimum of 256 MB and a maximum of 512 MB to the java process, add the following parameters to wrapper.cmd_ line:

   ```
   -Xms256M -Xmx512M
   ```

   The result will appear as follows:

   ```
   wrapper.cmd_line=$(wrapper.javabin) -Xms256M -Xmx512M -Djava.endorsed.dirs==$
   (ACTAHOME)\ext\webserver\common\endorsed ...
   ```

5. Save and close the file.

4.5.1.2 Increasing the java heap memory in UNIX

1. Stop the Data Services Web Server.
2. In the installation directory, navigate to /ext/WebServer/bin.
3. Using a text editor, open the setclasspath.sh file.

4. To allocate a minimum of 256 MB and a maximum of 512 MB to the java process, add the following lines to the end of the setclasspath.sh file:

```
JAVA_OPTS="-Xms256M -Xmx512M"
export JAVA_OPTS
```

5. Save and close the file.

4.5.2 Analysis options

The following table lists the types of objects that can appear in the navigation tree and provides a summary of what information appears in the right pane when you select that object type.

To view or hide a pane in the display such as an attributes pane, click the up/down arrow in its header bar.

Moving the cursor over an object displays a pop-up window with more information about that object; for example:

- Table: Data flow (if applicable), datastore, and owner
- Business Objects report: Depending on the report type, the pop-up window displays the CMS server name, the Business View name, or the Universe name, for example.
- Universe objects: CMS server, universe, and class

The following table lists the objects in the navigation tree, the corresponding tabs, and the content of each.

Object	Tab	Associated content
Repository	Overview	*Repository name* *Repository type*—The database type *Repository version*—The repository version number
Datastore	Overview	Overview information varies depending on the datastore type. The following entries apply to a datastore on Microsoft SQL Server. *Datastore type*—Database *Database type*—Microsoft_SQL_Server *User*—**<Database user name>** *Case sensitive*—Whether or not the database is case sensitive *Configuration*—**<The configuration selected in the datastore editor >** *SQL_Server version*—Microsoft SQL Server 2000 *Database name*—**<Database name >** *Server name*—**<The host computer name>**
Table	Overview	*Table name*—Table name *Datastore*—Datastore to which this table belongs

Object	Tab	Associated content
		Owner name—The table owner name in the database. *Business name*—Business-use name of the table if defined *Table type*—Table or template table *Last update date*—When the software last updated the table
	Impact	Graphically shows the end-to-end impact of the selected source table and the targets it affects. Clicking a table in the Impact diagram displays the object's attributes, which are the same as on the Overview tab.
	Lineage	Contains the same information as on the Impact tab except it describes the flow from target to source.
	Mapping tree	Displays the overall mapping information for each column in the table. Select between *Group by data flow* or *Group by column*.
Column	Overview	*Column name*—Column name *Table name*—Parent table of the column *Data type*—The data type for the column *Nullable*—yes/no *Primary key*—yes/no *Foreign key*—yes/no
	Impact	Graphically shows the end-to-end impact of the selected source column and the targets it affects. Clicking a column in the Impact diagram displays the object's attributes, which are the same as on the Overview tab.
	Lineage	Contains the same information as on the Impact tab except it describes the flow from target to source. It also displays any data flow objects associated with the column. Move the cursor over a data flow icon to display a pop-up window with the mapping information. Click on a data flow icon to open another pop-up window that displays the Auto Documentation information including mapping. Notice that in the auto documentation window, any objects that are not part of the lineage for this column are dimmed.
Universe	Overview	*Universe* name *Folder*—The folder name on the CMS server to where the Universe has been exported

Object	Tab	Associated content
		CMS server—The name of the CMS server for this Universe *Description*—The description created for the Universe in Designer
Business View	Overview	*Business View* name *Folder*—The folder name on the CMS server to where the Business View has been exported *Last update date*—When the Business View was last updated in the Business View Manager *CMS server*—The name of the CMS server for this Business View
Class	Overview	*Class* name *Universe*—Universe to which this class belongs *CMS server*—The name of the CMS server for this class
Object	Overview	*Object* name *Class*—The class to which this object belongs *Universe*—Universe to which this object belongs *CMS Server*—The name of the CMS server for this object *Source column*—The name of the source column for this object followed by the `<owner.table name>`
	Impact	Shows all of the reports that use the selected object
	Lineage	Shows column-level lineage for the selected object
Business Element	Overview	*Business element* name *Business view*—The Business View to which this business element belongs *CMS server*—The name of the CMS server for this object
Business Field	Overview	*Business field* name *Business element*—The Business Element to which this Business Field belongs *Business view*—The Business View to which this Business Field belongs *CMS server*—The name of the CMS server for this object
	Impact	Shows all the reports that use the field.
	Lineage	Shows column-level lineage for the selected Business Field.
Reports: Crystal	Overview	*Report* name

Object	Tab	Associated content
and Web Intelligence/ Desktop Intelligence		*Folder*—The folder on the CMS server where the report is stored *CMS server*—The name of the CMS server for this report *Last update date*—When the report was last updated
	Lineage	Shows the lineage to the Universe objects or Business View fields on which the report is based (if any) and the column-level lineage in the datastore.

Related Information

Auto Documentation Reports [page 1990]
Table-level and column-level analysis [page 1973]

4.5.3 Table-level and column-level analysis

Impact and lineage analysis at the table level shows datastores, tables, and reports/documents.

Impact and lineage analysis at the column level shows datastores, objects, fields, and reports/documents.

> **Note**
>
> If a report includes conditions predefined in a Universe, the columns used by those conditions do not appear on the Impact and Lineage Analysis report because the metadata collected for Universes is limited to the tables and columns referenced directly by Universe objects.

4.5.4 Impact and Lineage Analysis Settings control panel

The Impact and Lineage Analysis Settings control panel allows you to change the options for your reports. To open it, click *Settings* in the upper-right corner of the window.

4.5.4.1 Settings tab

The Settings tab allows you to change repositories. Select a repository from the drop-down list box and click *Apply*.

4.5.4.2 Refresh Usage Data tab

On the Refresh Usage Data tab, you can manually calculate column mappings.

4.5.4.2.1 Calculating column mappings

SAP Data Services can calculate information about target tables and columns and the sources used to populate them, for example for impact and lineage or auto documentation reports.

Calculating column mappings populates the internal ALVW_MAPPING view and the AL_COLMAP_NAMES table. The ALVW_MAPPING view provides current data to metadata reporting applications like Impact and Lineage Analysis. If you need to generate a report about a data flow that processes nested (NRDM) data, query the AL_COLMAP_NAMES table using a custom report.

Whenever a column mapping calculation is in progress, the Designer displays a status icon at the bottom right of the window. You can double-click this icon to cancel the process.

To calculate column mappings, you can:

- Enable the option in the Designer to automatically calculate column mappings.
- Execute the column mapping process manually from either the Designer or the Impact and Lineage Analysis application in the Management Console.

Related Information

Reference Guide: Metadata in Repository Tables and Views, Storing nested column-mapping data [page 1731]

4.5.4.2.1.1 To automatically calculate column mappings

To set the option to automatically calculate column mapping information, in the Designer select *Tools* > *Options* > *Designer* > *General* > *Automatically calculate column mappings*. This option is selected by default.

Note that if the Designer option *Automatically calculate column mappings* is cleared, any subsequent changes made to the data flow require that you manually recalculate the column mappings to ensure the ALVW_MAPPING view and the AL_COLMAP_NAMES table have the most current information.

4.5.4.2.1.2 To manually calculate column mappings

If the Designer option *Automatically calculate column mappings* is cleared and you want to generate reports, you can manually calculate the mappings. You can manually calculate column mappings at any time in either the Designer or the Management Console.

In the Designer, right-click in the object library and select ▶ *Repository* ▶ *Calculate column mappings* ◀.

In the Management Console:

1. Select *Impact and Lineage Analysis*.
2. Open the *Settings* control panel.
3. Click the *Refresh Usage Data* tab.
4. Select the Job Server that is associated with the repository you want to use.
5. Click *Calculate Column Mapping*.

On the Impact and Lineage Analysis *Overview* tab, you can expand *Data Flow Column Mapping Calculation* to view a list of data flows and the calculation status of each. If the mapping calculation is complete, the *Status* indicator is checked.

4.5.4.3 About tab

This tab provides SAP Data Services version information.

4.6 Operational Dashboard Reports

Operational dashboard reports provide graphical depictions of SAP Data Services job execution statistics. This feedback allows you to view at a glance the status and performance of your job executions for one or more repositories over a given time period. You can then use this information to streamline and monitor your job scheduling and management for maximizing overall efficiency and performance.

For more information about how to use these reports to answer performance questions, see the *Performance Optimization Guide*.

Related Information

Performance Optimization Guide: Measuring Performance [page 2106]

4.6.1 Dashboards home page

To view operational dashboards, on the Management Console home page, click *Operational Dashboard*.

The upper-right corner identifies the repository for the reports you are viewing. You can change the repository to view in the dashboards *Settings* control panel.

You can drill into these dashboards for more details. The navigation path at the top of the window indicates where you are in the operational dashboard reports hierarchy. Click the hyperlinks to navigate to different levels. Click the *Operational* tab to return to the dashboards home page.

There are two categories of dashboards: Job Execution Statistics and Job Execution Duration. Each category contains a current (snapshot) report and a historical (trend) report. Hence, the dashboards in the top row provide a snapshot for the last 24 hours, and the dashboards on the bottom row display trends over the last 7 days.

- Job execution statistics—Left side of dashboards home page. These reports depict job execution statistics—in general, how many jobs succeeded or had errors.
  - Current (snapshot) pie chart
  - Historical (trend) bar chart
- Job execution duration—Right side of page. These reports depict how long it took the jobs to run and whether those run times were within acceptable limits.
  Each report type includes a current (snapshot) speedometer and a historical (trend) line chart.

The following sections describe the contents of these reports and their subreports.

4.6.2 Job execution statistics

Job execution statistics display in two formats on the left side of the page:

- Current (snapshot) pie chart
- Historical (trend) bar chart

The color codes on these two charts apply to the status of the job's execution:

- Succeeded (green)
- One or more warnings (orange)
- One or more errors (red)
- Still running (blue)

4.6.2.1 Current (snapshot) pie chart

The pie chart displays status information for jobs that ran in the time period displayed. You can change the start time and execution time window in the dashboard Settings control panel.

The chart identifies the number and ratio of jobs that succeeded, had one or more warnings, had one or more errors, or are still currently running.

Click on the pie "slices" to drill into the report, which displays a table that shows the jobs in that status group. From there you can click on the status tabs to view the list of jobs in each group.

Each Job Execution Statistics table includes:

Option	Description
Repository name	The repository associated to this job.
Job name	The name of the job in the Designer.
Start time and End time	The start and end timestamps in the format hh:mm:ss.
Execution time	The elapsed time to execute the job.

Option	Description
System configuration	The name of the system configuration that applies to that job.

Related Information

Designer Guide: Datastores, Defining a system configuration [page 252]
Configuring the Job Execution Duration dashboard [page 1977]

4.6.2.2 Historical (trend) bar chart

The Job Execution Statistics History bar chart depicts how many jobs succeeded, had warnings, failed, or are still currently running on each of the last 7 days.

As with the Job Execution Statistics pie chart, you can click on the individual bars to drill into the report to display the Job Execution Statistics status tables (succeeded, warning, error, and so on) for a particular day.

Related Information

Current (snapshot) pie chart [page 1976]

4.6.3 Job Execution Duration

Job Execution Duration reports display in two dashboards on the right side of the dashboards home page:

- Current (snapshot) pie chart
- Historical (trend) line chart

These two charts describe whether all jobs in the selected repositories executed within the acceptable time frame that you established using the dashboard Settings control panel. This window is the amount of time that you allot to run your jobs so your target data warehouse is available to applications and users during business hours, for example.

4.6.3.1 Configuring the Job Execution Duration dashboard

The recommended process for setting up the Job Execution Duration dashboard is as follows.

1. In the Administrator, schedule your jobs for when you want them to run.

   Example: Schedule all jobs to start executing at 1:00 AM.

2. In the Operational Dashboard application, click *Settings* to open the control panel.
3. Select the repository to configure.
4. For the *View data starting at* value, enter the beginning of the time frame in which you want to view job execution as a whole on the speedometer dashboard (in the format HH:MM, from 00:00 to 23:59).

   For example, if all of your jobs begin to execute at 1:00 AM, it would be logical to set the *View data starting at* value to 1:00.
5. Enter a value for *Job execution time window* in number of hours (from 0 to 24) that you have available to run your ETL jobs.

   For example, if you want all of your jobs to finish executing by 5 AM, enter 4 hours, which would be the time window between 1 AM and 5 AM.
6. Optionally, add a value for the *Job execution time window tolerance* in increments of 1 hour.

   > Note
   >
   > The total of the *Job execution time window* value and the *Job execution time window tolerance* value cannot exceed 24.

   For example, suppose you determine that in some circumstances it is okay if some jobs finish executing by 6:00 AM. In this case, set the *Job execution time window tolerance* to 1 (extending the window from 5:00 AM to 6:00 AM).

Related Information

Management Console Guide: Batch Jobs, Scheduling jobs [page 1891]

4.6.3.2 Current (snapshot) speedometer

The speedometer graphic displays the current window of time allotted for your jobs to complete their executions and whether or not the jobs as a whole finished within the selected job execution time window.

> Note
>
> This dashboard does not indicate whether a job succeeded or had errors; it only displays execution duration. Drill into the speedometer dashboard to see a table that displays whether or not the job succeeded and if and where it finished executing within the job execution time window.

Continuing with the example described in the preceding process, the following morning, after running your jobs overnight, you want to verify that they executed within the time window of 1 AM to 5 AM. Open the Operational Dashboard application and view the Job Execution Duration speedometer dashboard.

The speedometer needle points in the green zone, which represents the *Job execution time window* setting in the Settings control panel. In the example, that setting was 4 hours (from 1:00 AM to 5:00 AM). Therefore, all jobs in the repository finished executing within the desired time frame.

The solid red bar indicates when the first job actually started, which in the case of the example was 1:00 AM.

The time below the needle hub indicates when the last job finished executing for the selected repository. If the start time of the first job execution and the *View data starting at* value are the same, as in the example, that value equates to the total execution time for all the jobs (1:15 in the example).

If, however, you entered a *View data starting at* value other than when the first job actually started, to calculate the overall job execution duration period, subtract the time the first job started (indicated by the red bar and noted below the dashboard) from the time below the needle hub.

The time period displayed above the speedometer graphic indicates the viewing period. For example,

3/6/07 1:00 AM - 3/6/07 2:57 PM

indicates that you are viewing a dashboard that reflects all jobs that ran between 1:00 AM and 2:57 PM. The time 1:00 AM is the same as the *View data starting at* value. The time 2:57 PM is when you opened the Operational Dashboard control panel, which acts as a time stamp for the end period of the data you are actually viewing.

Therefore, the color zones on both Job Execution Duration dashboards (the speedometer and the history line chart) represent:

- Green (normal)—All jobs executed within the job execution time window. Specify this window with the *Job execution time window* setting. The green range of the speedometer represents this value.
- Yellow (warning)—At least one job finished executing during the tolerance period. Tolerance is the amount of time that jobs can run beyond your normal *Job execution time window*. Specify this tolerance limit with the *Job execution time window tolerance* setting. This value is represented in the yellow range of the speedometer and in line charts.
- Red (exceeded)—At least one job finished executing beyond the normal or warning (tolerance) limits.

4.6.3.2.1 Viewing job execution duration details

Click the speedometer dashboard to drill into the report and display a table of the Job Execution Duration for all jobs.

The Job Execution Duration table displays the repository name, the job name (for successful jobs, click the job name to drill down to the Data Flow Execution Duration table), the start and end time of the job execution, the execution time (elapsed), the status, and whether the job is associated with a system configuration.

4.6.3.2.2 Viewing data flow execution duration details

From the *Job Execution Duration details* window, you can drill down to a job to view its data flow execution duration times and audit details if applicable. Click on a job name to display the Data Flow Execution Duration.

The Data Flow Execution Duration table shows the repository name, the data flow name, the job name, the start and end time of the data flow execution, the elapsed execution time, the number of rows extracted and loaded, and the audit status.

If you enabled auditing for the job, the Audit status column displays one of the following statuses:

- Not audited
- Success: All audit rules succeeded

- Collected:This status occurs when you define audit labels to collect statistics but do not define audit rules
- Failed: Audit rule failed

You can view the Audit Details page from the Table tab by clicking a value in the Audit status column.

4.6.3.2.3 Viewing audit details

If auditing has been enabled, on the Data Flow Execution Duration table, click the audit status for the data flow to display the Audit Details page.

In addition to job name, data flow name, and job start time, the report includes:

- Audit rule information, if applicable
- Audit failed information, if applicable
- A table with the audit labels and values

Related Information

Designer Guide: Data Assessment, Using Auditing [page 452]

4.6.3.3 Historical (trend) line chart

The Job Execution Duration History line chart shows the trend of job execution performance over the last 7 days. The vertical axis is in hours, the horizontal axis shows the date, and the colored dots indicate the overall results of job execution duration based on the settings defined in the Settings control panel.

Each point denotes the Job Execution Duration (total elapsed time) of all the jobs for that day.

Click on the points to drill in to display the Job Execution Duration table.

Also as with the speedometer dashboard, you can drill into the job for data flow execution duration times and audit details by clicking on a job name to display the Data Flow Execution Duration window. Additionally, if auditing has been enabled, click on the audit status to display the Audit Details page.

Click the *View all history* button to display two tables:

- *Job Execution History*: This table lets you view execution history for all jobs or individual jobs. You can also select over how many days you want to view the history.
  This table displays the same information as in the Job Execution Duration table: repository name, job name, the start and end time of the job execution, the execution time (elapsed), the status, and whether the job is associated with a system configuration.
  - Click the Trace, Monitor, or Error link to open the Log Viewer page, which displays the logs associated with the job. Click the tabs to view the other types of logs.
  - Click the Performance Monitor link to open the Performance Monitor page.
- *Data Flow Execution History*: This table includes three options for customizing the display:

- *Data Flow*: Enter a data flow name for which to search and click *Search*.
- *Job Name*: Select all jobs or an individual job.
- *View history for x days*: Select over how many days you want to view the history

This table displays almost the same information as in the Data Flow Execution Duration table: repository name, data flow name, job name, the start and end time of the data flow execution, the execution time (elapsed), and the number of rows extracted and loaded.

Related Information

Configuring the Job Execution Duration dashboard [page 1977]
Current (snapshot) speedometer [page 1978]

4.7 Data Validation Dashboard Reports

Data Validation dashboard reports provide graphical depictions that let you evaluate the reliability of your target data based on the validation rules you created in your SAP Data Services batch jobs. This feedback allows business users to quickly review, assess, and identify potential inconsistencies or errors in source data.

4.7.1 Configuring Data Validation dashboards

To generate meaningful Data Validation dashboards, follow this process:

1. In your jobs in Designer, create data flows that contain validation transforms with validation rules.

   You use validation transforms to:

   - Verify that your source data meets your business rules.
   - Take the appropriate actions when the data does not meet your business rules.
2. In the Data Validation application, create functional areas. A functional area is a virtual group of jobs that relate to the same business function, for example Human Resources or Customers. Functional areas can contain multiple jobs and one or more data validation business rules.
3. In the Data Validation application, create business rules. Business rules are typical categories of data, for example Social Security Number or Address.

   These business rules contain validation rules that you created in your validation transforms in your data flows.

Begin by opening the Management Console and clicking the *Data Validation* name or icon link on the Home page.

Related Information

Creating functional areas [page 1982]

Creating business rules [page 1983]

Designer Guide: Data Assessment, Using the Validation transform [page 450]

Reference Guide: Transforms, Validation transform [page 1472]

4.7.1.1 Creating functional areas

After you create data flows with validation transforms and rules in Designer, next create functional areas.

> i Note
>
> If you do not create any functional areas, the dashboard displays statistics for all jobs for the selected repository.

4.7.1.1.1 To create functional areas

1. In the Data Validation module, click the *Settings* link.

   The *Repository* tab displays.
2. Verify the desired repository displays.

   To change the repository, select a different one from the drop-down list and click *Apply*.
3. Click the *Functional area* tab.

   The selected repository displays.
4. Type a name for the new functional area (such as `Customer`) and optionally a description.
5. Click *Save*.

   A list of batch jobs (and the associated system configuration for each, if any) appears that lets you select the jobs you want to include in this functional area.

   You can change the sort order of the table by clicking the arrow in a column heading.
6. From the list of Available batch jobs, select the check box for each job to include in this functional area and click the arrow button to move it to the *Associated batch jobs* window.
   - Jobs are not reusable among functional areas.
   - In the Administrator, deleting job information on the Batch Job Status page (Batch Jobs History) also clears data validation statistics from Data Validation Metadata Reports.
7. Click *Apply* to save the changes and keep the Functional area window open, for example to add more functional areas.

   Clicking *OK* saves your changes and closes the Settings control panel.

   Clicking *Cancel* closes the Settings control panel without saving changes.

To add another functional area, on the Functional area tab click *Add* and follow steps 4 through 7 in the previous procedure.

To display a different functional area for editing, select it from the drop-down list.

Delete a functional area by clicking *Delete* next to the functional area selected in the drop-down list.

Next, create business rules to associate the validation rules in your data flows with your functional areas.

Related Information

Designer Guide: Datastores, Creating and managing multiple datastore configurations [page 239]

4.7.1.2 Creating business rules

After creating functional areas, associate business rules to each functional area as follows.

> Note
>
> If you do not create business rules, each validation rule in the jobs that you have associated with a functional area becomes its own business rule.

4.7.1.2.1 To create business rules

1. In the Settings control panel, click the *Business rule* tab.

   On the *Business rule* tab, the default repository displays. (To change the repository, click the *Repository* tab, select a different repository, and click *Apply*.)
2. From the drop-down list, select the functional area to which this business rule will belong.
3. In the business rule definition area, type a name for the business rule such as *Phone number*.
4. Select a priority for how significant this group will be to your business end users: High, Medium, or Low.
5. Type an optional description for the business rule such as *Phone and FAX*. In this example, the validation rule checks to see if telephone and fax numbers in the USA conform to the seven-digit standard.
6. Click *Save*.

   A list of validation rules appears. Select the validation rules you want to include in this business rule. See the table below for a description of the columns in the lists of validation rules (scroll horizontally to view the other columns).
7. From the list of *Available validation rules*, select the check box for all of the rules to include in this business rule and click the arrow button to move it to the Associated validation rules pane.

   > Note
   >
   > Validation rules are not reusable among business rules (you can use a validation rule only once and only in one business rule).
8. Click *Apply* to save the changes and keep the *Business rule* window open, for example to add more business rules.

Clicking *OK* saves your changes and closes the Settings control panel.

Clicking *Cancel* closes the Settings control panel without saving changes.

The columns in the validation rule lists are as follows. Scroll horizontally to view the other columns. Note that you can change the sort order of the tables by clicking the arrow in a column heading.

Column	Description
Validation rule name	The validation rule name. The default is the column name, unless you create a new name in the Validation transform *Properties* dialog box.
Description	The description for the validation rule.
Full path	Describes the hierarchy in the software from the job level to the data flow level to indicate where this validation rule has been defined.
System configuration	The name of the system configuration selected for this job.

To add another business rule, on the *Business rule* tab click *Add* and follow steps 3 through 8 in the previous procedure.

To display a different functional area so you can edit its business rules, select it from the *Functional area* drop-down list.

To display a different business rule for editing, select it from the *Business rule* drop-down list.

Delete a business rule by clicking *Delete* next to the business rule selected in the drop-down list.

Related Information

Designer Guide: Data Assessment, Using the Validation transform [page 450]
Reference Guide: Transforms, Validation transform [page 1472]
Designer Guide: Datastores, Creating and managing multiple datastore configurations [page 239]

4.7.1.3 Enabling data validation statistics collection

To enable data validation statistics collection for your reports, you must verify two options—one at the validation transform level and one at the job execution level.

4.7.1.3.1 Validation transform level

In Designer, navigate to the validation transforms from which you want to collect data validation statistics for your reports. For the columns that have been enabled for validation, in the transform editor click the *Validation transform options* tab and select the check box *Collect data validation statistics*.

4.7.1.3.2 Job execution level

When you execute a job, the Execution Properties window displays. On the Parameters tab, clear the option *Disable data validation statistics collection* (the default) before executing the job.

To execute a job without collecting statistics, select the *Disable data validation statistics collection* option, which suppresses statistics collection for all the validation rules in that job.

4.7.2 Viewing Data Validation dashboards

To view Data Validation dashboards, on the Management Console Home page, click *Data Validation*. The functional area view displays.

The upper-right corner identifies the repository for the reports you are viewing. You can change the repository to view in the Settings control panel.

Some dashboards let you to drill in to some of the components for more details. The navigation path at the top of the window indicates where you are in the Data Validation dashboard reports hierarchy. Click the hyperlinks to navigate to different levels.

Related Information

Functional area view [page 1985]

Repository tab [page 1990]

4.7.3 Functional area view

The top-level Data Validation dashboard view is the functional area view. This view includes two dashboards:

Dashboard	Description
Functional area pie chart	A current (snapshot) report.
History line chart	A historical (trend) report that displays trends over the last 7 days. To display this chart, click the link below the pie chart.

To change the functional area dashboards to view, select a different functional area from the drop-down list at the top of the window. Select *All* to view data for all batch jobs in that repository.

If there are no functional areas defined, the chart displays statistics for all jobs for the selected repository.

Related Information

Functional area pie chart [page 1986]
History line chart [page 1986]

4.7.3.1 Functional area pie chart

The pie chart displays status information for jobs that ran in the time period displayed. The data collection period begins at the time set on the Repository tab in the Settings control panel and ends with the time you open the report Web page.

The color codes on this chart apply to the volume of validation rules that passed or failed:

- Green—The percentage of rules that passed for the selected functional area.
- Red—The percentage of rules that failed for the selected functional area with a High priority label.

> Note
>
> If you have All selected for Functional area, the failed portion appears red and does not include priority information.

- Orange—The percentage of rules that failed for the selected functional area with a Medium priority label.
- Blue—The percentage of rules that failed for the selected functional area with a Low priority label.

Click on the failed pie "slices" to drill into the report:

- If there is at least one functional area and business rule defined, the Business rule view displays.
- If no functional areas or business rules are defined, the Validation rule view displays.

Related Information

Repository tab [page 1990]
Creating business rules [page 1983]
Business rule view [page 1987]
Validation rule view [page 1987]

4.7.3.2 History line chart

The History line chart displays the percentage of all validation rule failures that occurred on each of the last seven days. The software collects the number of failures for the last run of all of the batch jobs that are associated with this functional area on each day.

To display the History line chart, click the link below the pie chart.

4.7.4 Business rule view

The Business rule dashboard is a bar chart that shows the percentage of validation failures for each defined business rule. You access this chart by clicking on one of the failed "slices" of the functional area pie chart.

> ℹ Note
>
> If no functional areas or business rules have been defined, drilling down on the functional area pie chart displays the Validation rule view. Therefore, the "business rules" are the same as the validation rules.

The data collection period begins at the time set on the Repository tab in the Settings control panel and ends with the time you open the report Web page.

You can filter the view by selecting an option from the Priority drop-down list:

- All
- High priority
- Medium priority
- Low priority

The chart displays 10 bars per page; click the *Next* or *Previous* links to view more.

To view the business rules associated with another functional area, select it from the Functional area drop-down list.

To see the validation rules that apply to a particular business rule, click on that bar in the chart. The Validation Rule View displays.

Related Information

Repository tab [page 1990]
Validation rule view [page 1987]

4.7.5 Validation rule view

When you view the Functional area pie chart (the Data Validation Home page) and click on one of the failed pie "slices" to drill into the report, if no functional areas or business rules have been defined, the Validation rule view displays.

The Validation Rule page includes two dashboards:

Dashboard	Description
Validation rule bar chart	A current (snapshot) report.
History line chart	A historical (trend) report that displays trends over the last 7 days. To display this chart, click the link below the Validation rules bar chart.

The chart displays 10 bars per page; click the *Next* or *Previous* links to view more.

To view the validation rules for a different business rule, select it from the *Business rule* drop-down list.

Related Information

Validation rule bar chart [page 1988]
History line chart [page 1988]

4.7.5.1 Validation rule bar chart

The Validation rule bar chart displays the percentage of rows that failed for each validation rule.

The Data collection period begins at the time set on the Repository tab in the Settings control panel and ends with the time you open the report Web page.

You can click on a bar to display the Sample data view.

Related Information

Repository tab [page 1990]
Sample data view [page 1988]

4.7.5.2 History line chart

The History line chart displays the percentage of all validation rule failures that occurred on each of the last seven days. The software collects the number of failures for the last run of all of the batch jobs that are associated with this functional area.

To display this chart, click the link below the Validation rules bar chart.

4.7.6 Sample data view

If you have enabled the option to collect sample data in a validation transform, the Sample data page includes two images:

- *Sample data table* [page 1989]
- *History line chart* [page 1989]

If you have not configured the validation transform to collect sample data, only the History line chart displays.

To view sample data from another validation rule, select it from the drop-down list.

Related Information

Reference Guide: Transforms, Validation transform [page 1472]

4.7.6.1 Sample data table

If you have configured the validation transform to collect sample data , this page displays up to 50 rows of sample data in tabular form.

The column that maps to the currently selected validation rule appears highlighted.

Related Information

Reference Guide: Transforms, Validation transform [page 1472]

4.7.6.2 History line chart

The History line chart displays the percentage of all validation rule failures that occurred on each of the last seven days. The software collects the number of failures for the last run of all of the batch jobs that are associated with this functional area.

To display this chart, click the link below the sample data table.

4.7.7 Data Validation dashboards Settings control panel

The dashboard settings window allows you to change the options for your reports.

In the upper-right corner of the window, click *Settings*.

The three settings tabs are:

- *Repository tab* [page 1990]
- *Functional area tab* [page 1990]
- *Business rule tab* [page 1990]

For all of the Settings pages:

- Click *Apply* to save the changes and keep the Business rules dialog box open, for example to add more business rules.

- Clicking *OK* saves your changes and closes the Settings control panel.
- Clicking *Cancel* closes the Settings control panel without saving changes.

4.7.7.1 Repository tab

The settings on the Repository tab include:

- *Repository*—Select a repository to view from the drop-down list.
- *View data starting at*—Enter the time from when you want start viewing data in the format HH:MM (from 00:00 to 23:59). On the Data Validation dashboard charts, the end of the time window is the moment you open one of the dashboard Web pages.
  For example, entering 02:00 means your data validation reports will display data from the repository starting at 2:00 a.m. through the time you open the report page. Each dashboard page displays the time frame for which the dashboard statistics apply.

4.7.7.2 Functional area tab

For details on how to use the Functional area tab settings, see *Creating functional areas* [page 1982].

4.7.7.3 Business rule tab

For details on how to use the business rule tab settings, see *Creating business rules* [page 1983].

4.8 Auto Documentation Reports

Auto Documentation reports provide a convenient and comprehensive way to create printed documentation for all of the objects you create in SAP Data Services. Auto Documentation reports capture critical information for understanding your jobs so you can see at a glance the entire ETL process.

After you create a project, you can use Auto Documentation reports to quickly create a PDF or Microsoft Word file that captures a selection of job, work flow, and/or data flow information including graphical representations and key mapping details.

The types of information provided by Auto Documentation reports include:

- Object properties—Apply to work flows, data flows, and ABAP data flows
- Variables and parameters—Apply to jobs, work flows, data flows, and ABAP data flows
- Table usage—Shows all tables used in this object and its child objects.
  For example for a data flow, table usage information includes datastores and the source and target tables for each

- Thumbnails—An image that reflects the selected object with respect to all other objects in the parent container. Applies to all objects except jobs.
- Mapping tree—Applies to data flows

The next section includes object descriptions that describe these components in more detail as they apply to that object.

4.8.1 Navigation

Auto Documentation reports allow you to navigate to any project, job, work flow, data flow, or ABAP data flow created in the Designer and view information about the object and its components.

To open Auto Documentation reports, from the Management Console Home page, click *Auto Documentation*.

Auto Documentation navigation is similar to that of Impact and Lineage reports. The Auto Documentation page has two primary panes:

- The left pane shows a hierarchy (tree) of objects.
- The right pane shows object content and context.

Expand the tree to select an object to analyze. Select an object to display pertinent information in the right pane. In the right pane, select tabs to display desired information. Tabs vary depending on the object you are exploring.

Note that if the subcategory within a tab has a window icon on the far right side of its header bar, you can collapse or expand that part of the display.

Related Information

Impact and Lineage Analysis Reports [page 1967]

4.8.1.1 Searching for a specific object

1. Type the object name in the *Objects to analyze* search field. The search field is not case sensitive, you may use the percent symbol (%) as a wildcard, and spaces are allowed.
2. Click the search icon (binoculars) or press Enter.

The top level of the navigation tree displays the current repository. You can change this repository in the Settings control panel.

When you first open Auto Documentation, in the right pane the Project tab displays with hyperlinks to the projects in the repository. You can click on these links to drill into the project to display a list of jobs. You can also continue to click the job name links to continue drilling into objects for more information (thumbnails, object properties, and so on).

You can also navigate to this content by selecting objects in the navigation tree. The hierarchy of the tree matches the hierarchy of objects created in Designer.

- Repository
- Project
- Job
- Work flow
- Data flow

4.8.1.2 Repository

Clicking a repository name in the navigation tree displays the Overview tab in the right pane, which includes the following:

- Repository name—The name given to the repository in the Repository Manager
- Repository type—The type of repository such as the database type
- Repository version—The repository version

4.8.1.3 Project

Clicking on a project name in the navigation tree displays the Project tab in the right pane, which displays the list of jobs in the project. You can click on these jobs to drill into the hierarchy.

4.8.1.4 Job

Clicking on a job name displays information on two tabs:

- *Name of job*—This tab includes:
  - Description
  - A graphical image of the next level of objects in the job (for example one or more work flows or data flows)
  - Variables and parameters used in the job
- *Table usage*—This tab lists the datastores and associated tables contained in the selected object if applicable.

4.8.1.5 Work flow

Clicking on a work flow name displays information on two tabs:

- *Name of work flow*—This tab includes:
  - Thumbnail—This image reflects the selected object with respect to all other objects in the parent container.
    For example, if this work flow is the only one in the job, it appears alone; but if there are two other work flows in the job, they will also appear in the thumbnail image. You can click the other thumbnail images to navigate to them, which replaces the content in the right pane with that of the selected object.

    - A graphical image of the object's workspace window as it appears in the Designer.
      You can click on objects to drill into them for more details. Objects that have a workspace in Designer will display that workspace.
    - Object properties—Displays properties such as *Execute only once* or *Recover as a unit* if set.
    - Variables and parameters used in the work flow
- *Table usage*—This tab lists the datastores and associated tables contained in the selected object if applicable.

4.8.1.6 Data flow

Clicking a data flow name displays information on three tabs:

- *Name of data flow*—This tab includes:
    - Thumbnail—This image reflects the selected object with respect to all other objects in the parent container.
    - A graphical image of the object's workspace window as it appears in Designer.
      You can click on an object in the data flow to drill into it for more information:
      A table displays the following information when selected:
        - A thumbnail of the table with respect to the other objects in the data flow
        - Table properties including Optimizer hints (caching and join rank settings) and Table basics (datastore and table name)

      A transform displays the following information when selected:
        - A thumbnail of the transform with respect to the other objects in the data flow
        - Mapping details including column names, mapping expressions, descriptions, and data types.

      To go to the top level of the data flow object, click the data flow name in the navigation tree, or click the Back button on your browser.
    - Object properties—Data flow properties such as *Execute only once* and *Degree of parallelism*
    - Variables and parameters used in the job
- *Mapping tree*—This tab displays a list of target tables. You can expand or collapse each table display by clicking its header bar. Each target table lists its target columns and the mapping expressions (sources) for each.
- *Table usage*—This tab lists the datastores and associated tables contained in the selected object if applicable.

4.8.2 Generating documentation for an object

For most objects, you can quickly generate documentation in Adobe PDF or Microsoft Word format by clicking the printer icon next to the object name in the right pane.

4.8.2.1 Printing Auto Documentation for an object

1. Select the highest-level object you want to document in the left pane.

2. In the right pane, click the printer icon next to the object name.
3. In the Print window, select the check boxes for the items to include in your printed report.
4. Select PDF or Microsoft Word format.
5. Click *Print*.

   The Windows File download dialog box displays. Click Open to generate and open the file now, or click Save to save the file to any location.
6. After saving or printing your report, click *Close* to close the Print window.

4.8.3 Auto Documentation Settings control panel

The Auto Documentation Settings control panel allows you to change the options for your reports.

In the upper-right corner of the window, click *Settings*.

The following options are available in the *Settings* tab:

- *Repository*—Select a repository from the drop-down list box and click *Apply*.
- *ImageDisplay*—Specify whether a whole or partial image is displayed for large images. Setting the option to *Partial* improves performance and memory utilization.

The *About* tab provides software version information.

4.9 Data Quality Reports

Many Data Quality transforms generate information about the data being processed. Data Quality reports provide access to that data processing information. You can view and export these Crystal Reports for batch and real-time jobs. The statistics-generating transforms include Match, USA Regulatory Address Cleanse, Data Cleanse, Global Address Cleanse, Geocoder, and DSF2 Walk Sequencer transforms. Report types include addressing reports, transform-specific reports, and transform group reports.

> i Note
>
> Viewing or exporting data quality reports requires that you have first installed the Data Services APS services on the same machine that contains the BI platform (or IPS) Central Management Server. See the *Data Services Installation Guide* and *Master Guide* for more information.

Configuring report generation

To enable report generation, ensure the option *Generate report data* in the transform editor is enabled, or use substitution variables to set this and other transform options at a repository level.

For details about setting the options for transforms, see the *Reference Guide*. For details about using substitution variables, see the *Designer Guide*.

The following table lists the available reports and their associated transforms.

Report	Transform(s)
US CASS report: USPS Form 3553	USA Regulatory Address Cleanse
NCOALink Processing Summary report	USA Regulatory Address Cleanse
US Addressing report	USA Regulatory Address Cleanse
US Regulatory Locking report	USA Regulatory Address Cleanse
Delivery Sequence Invoice report	DSF2 Walk Sequencer
Canadian SERP report: Statement of Address Accuracy	Global Address Cleanse
Australia AMAS report: Address Matching Processing Summary	Global Address Cleanse
Address Information Codes Sample report	Global Address Cleanse and USA Regulatory Address Cleanse
Address Information Code Summary report	Global Address Cleanse and USA Regulatory Address Cleanse
Address Validation Summary report	Global Address Cleanse and USA Regulatory Address Cleanse
Address Type Summary report	Global Address Cleanse and USA Regulatory Address Cleanse
Address Standardization Sample report	Global Address Cleanse and USA Regulatory Address Cleanse
Address Quality Code Summary report	Global Address Cleanse
Data Cleanse Information Code Summary report	Data Cleanse
Data Cleanse Status Code Summary report	Data Cleanse
Best Record Summary report	Match
Match Contribution report	Match
Match Criteria Summary report	Match
Match Source Stats Summary report	Match
Match Duplicate Sample report	Match
Match Input Source Output Select report	Match
Match Multi-source Frequency report	Match
Geocoder Summary report	Geocoder

Opening and viewing reports

To view Data Quality reports, from the Management Console Home page, click *Data Quality Reports*. The *Batch job reports* tab appears.

Note

To view a Data Quality report, after clicking it, you will need to enter a user name and password for the Data Services repository. Once you have logged in, you may view as many Data Quality reports as you want during a single session in the Management Console without needing to provide the password again. This is the default behavior. See the *Administrator Guide* for more information, including how to change this behavior.

ℹ Note

Reports require the Arial Unicode MS font. This font is designed to support a wide variety of code pages and is included with Microsoft Office 2002 and later. To view Unicode data in PDF reports, this font must be installed on the computer being used to view the reports. For more information about the Arial Unicode MS font, including how to install from a Microsoft Office CD, visit the Microsoft Web site.

Related Information

Reference Guide: Transforms, Global Address Cleanse [page 1197]
Reference Guide: Transforms, USA Regulatory Address Cleanse [page 1317]
Reference Guide: Transforms, DSF2 Walk Sequencer [page 1172]
Designer Guide: Variables and Parameters, Overview of substitution parameters [page 414]
Designer Guide: Multiple data source statistics reporting [page 679]
Administrator Guide: Password protection for a Data Services repository [page 45]

4.9.1 Lists of available reports

After opening the Data Quality Reports module in the Data Services Management Console, the *Batch job reports* tab displays a list of jobs and their associated available reports. Click the *Real-time job reports* tab to display reports available for real-time jobs. The *Real-time job reports* tab includes the same information that is on the *Batch job reports* tab.

The upper-right corner of either the Batch or Real-time pages displays the repository for the reports you are viewing. You can change the repository that you want to view in the *Settings* control panel.

You can filter the list of reports displayed by selecting a job name and/or a date or date range.

To filter by job, select the job name from the *Job name* drop-down menu. Or type the name, or type part of the name and a wildcard character (% or *), into the wildcard search string box and click *Search*. The Search field is not case sensitive and spaces are allowed.

To filter by when the job(s) executed, select one of the following options:

- Show last execution of a job.
- Show status relative to today: Select the number of previous days over which to view job executions.
- Show status as a set period: Type the date range or select the dates by clicking the calendar icons.

Click *Search* to update the list.

The report list includes the following headings. You can sort or reverse sort on any column by clicking the arrow next to the column name.

- Report. Click the icon in this column to go to a page that lists the reports for the associated job.

- Job name. The name of the job in the Designer.
- Status. The execution status of the job: green (succeeded), yellow (had one or more warnings), red (had one or more errors), or blue (still executing).
- Start time and End time. The start and end dates (in the format yyyy-mm-dd) and times (in the format hh:mm:ss).
- Execution time. The elapsed time to execute the job.
- Rows extracted and Rows loaded. The number of rows read by the job and the number of rows loaded to the target.

Related Information

Data Quality Reports Settings control panel [page 1998]

4.9.2 List of reports by job

On the list of available reports page (either the *Batch job reports* or *Real-time reports* tabs), click an icon in the Report column to display a page with the list of reports that are available for the associated job. The job name displays at the top of the page.

Filter the list by performing one of the following actions:

- Select a report name from the drop-down list. This action automatically populates the Search box and submits the request.
- Type the name or part of the name and a wildcard character (% or *) into the Search box and click *Search*. The Search field is not case sensitive and spaces are allowed.

The list of reports includes the following column headings. You can sort or reverse sort on any column by clicking the arrow next to the column name.

- Report. Click the icon in this column to open the report, which displays in a new browser window. The three icons represent the type of report:
  - Summary
  - Transform-specific report
  - Report for a group of transforms (match set or associate set)
- Path name. The transform location within the job if the report comes from a transform, match set, or associate set. The location displays in the form of <data flow name>/<transform name>. If the report comes from a job summary report, the job name displays.
- Report name. The type of report (for example Address Type Summary report or Match Contribution report).
- Object name. The source object used for the report, typically the transform name used in the data flow (blank for job summary reports).

4.9.3 Data Quality Reports Settings control panel

Use the Settings control panel to select a different repository for viewing reports. In the upper-right corner of the window, click *Settings*. Select a repository from the drop-down list box and click *Apply*.

4.9.4 Report options

After opening a report, use the toolbars at the top (and bottom) of the report window to perform the following tasks:

- Export to one of the following formats: Crystal Report (RPT), Adobe Acrobat (PDF), Microsoft Word - Editable (RTF), or Rich Text Format (RTF). The Export Report dialog box also lets you select a page range to export.
- Print the full report or a range of pages
- Show the group tree, which displays a hierarchical navigation tree pane. Select links in the tree to navigate to different sections of the report.
- Navigate through the report using the page links and fields at the top and bottom of the report page
- Refresh the report
- Search for text
- Resize the view

In many of the reports, you can select the transform name, path, charts, or other components to display a separate report that is specific to that component. To identify which components are enabled for viewing as a subreport, move the cursor over the report and look for objects that make it change to a hand icon. The description (name) of the newly displayed report then appears in the tab at the top of the report. The tabs displays all the subreports you have accessed. Select *Main Report* to return to the original report.

4.9.5 Troubleshooting reports

The following are several report tips and solutions that you can use to troubleshoot report problems.

Font

Reports require the Arial Unicode MS font. This font is designed to support a wide variety of code pages and is included with Microsoft Office 2002 and later. To view Unicode data in PDF reports, this font must be installed on the computer being used to view the reports. For more information about the Arial Unicode MS font, including how to install from a Microsoft Office CD, visit the Microsoft Web site.

Slow reports or timeout error

If you encounter a timeout error or notice that your reports are running appreciably slower, your DBA should update the database statistics on your repository database, primarily the AL_STATISTICS ADDRINFOCODEDATA ADDRSTATUSCODEDATA table. In Oracle, this is called "Gather statistics."

If you use an Oracle database, here are some additional options that your DBA can modify to try to correct the issue (other databases may have similar options):

- Increase the Processes value to 400 or greater.
- Increase the PGA Memory to 194MB or greater.
- Set the Cursor_Sharing value to Force.

4.9.6 USA CASS report: USPS Form 3553

Description

The Coding Accuracy Support System (CASS) report is a facsimile of United States Postal Service (USPS) Form 3553. You need this form to qualify mailings for postage discounts. For more information about CASS, visit the USPS RIBBS (Rapid Information Bulletin Board) web site at *https://ribbs.usps.gov/* .

To enable the report

To generate this report, use the USA Regulatory Address Cleanse transform. Ensure the following options have been defined, or configure the appropriate substitution parameters:

- Enable ▶ *Report And Analysis* ▶ *Generate Report Data* ◀.
- Disable ▶ *Non Certified Options* ▶ *Disable Certification* ◀.
- Set all applicable options in the *CASS Report Options* option group.
- Set all applicable options in the *Assignment Options* group to **<Yes>** and specify a valid directory path for each. Applicable options are those that are required for CASS.

Related Information

Designer Guide: Data Quality, Beyond the basic address cleansing, Multiple data source statistics reporting [page 679]

4.9.7 NCOALink Processing Summary Report

Description

The NCOALink Processing Summary Report can provide a detailed breakdown of the various codes returned by NCOALink and ANKLink processing, information regarding how the NCOALink job was configured, and summary information for some job processing statistics. The report generates information and completes the applicable fields based on your service provider level. The NCOALink Processing Summary Report can be used for USPS certification and audit purposes.

To enable the report

To generate this report, use the USA Regulatory Address Cleanse transform. On the *Options* tab, enable the following options or configure the appropriate substitution parameters:

- Set ▶ *Assignment Options* › *Enable NCOALink* ◀ to Yes.
- Set ▶ *Report And Analysis* › *Generate Report Data* ◀ to Yes.
- Ensure that the other required NCOALink fields on the Options tab have been correctly configured.

How to read this report

The first section of the report includes the report title, the date of the report, and other general information about your job as follows:

- The job name
- The run ID, which specifies a unique identification of an instance of the executed job
- The name of the repository where the job is located
- The official transform name
- The path to the transform in the form <data flow name>/<transform name>

Depending on the transform option settings, the remainder of the report includes the following sections:

- Move Update Summary: Contains information about the job configuration and statistics for move-updated addresses as well as pre-move update counts for Postcode2, DPV, LACSLink, and SuiteLink matches.
  The *Data Returned* value shows the processing mode for the mailing list. The possible values are *C* (change of address), *F* (return codes), or *S* (statistics). For a detailed description of the values, see the *Reference Guide*.
  To change the processing mode, modify the ▶ *NCOALink* › *Processing Options* › *List Processing Mode* ◀ option in the transform.
  The *Match Logic* value is a single character code that denotes the combination of move types processed. The setting for each move type (business, individual, and family) displays in the area of the report immediately below the list of all processes used. To change the match logic, modify the ▶ *NCOALink* › *Processing Options* › *Retrieve Move Types* ◀ option in the transform.
  The *Mail Class* character code is derived from the settings for processing standard mail, first class mail, package services, and periodicals. The setting for each mail class is shown in the section immediately below

the list of all processes used. To change the settings, modify the previously noted fields located in the *NCOALink* > *Processing Options* option group in the transform.

- NCOALink Move Type Summary: Displays statistics for the types of moves (Individual, Family, or Business) processed.
- NCOALink Return Code Summary: Displays a summary of the NCOALink codes returned.
- Move Effective Date Distribution Analysis: Contains the number of records in each category and time period. Categories include new address provided, new address not available, and new address not provided. Time frames include in months: 0-3, 4-6, 7-12, 13-18, 19 plus.
- ANKLink Return Code Summary: If ANKLink is enabled, displays a summary of the ANKLink codes returned. The section won't display if there aren't any ANKLInk return codes in the output.
- Return Code Descriptions: If enabled in the *NCOALink* > *Report Options* > *Generate Return Code Descriptions* option, displays detailed return code descriptions.
- Service Provider Summary: Contains information specific to processing for limited or full service providers. This section does not display if processing the job as an end user.

Related Information

Management Console Guide: Administrator, Exporting NCOALink certification logs [page 1878]
Reference Guide: Transforms, NCOALink options [page 1326]
Designer Guide: Data Quality, Multiple data source statistics reporting [page 679]

4.9.8 Delivery Sequence Invoice report

Description

The DSF2 Delivery Sequence Invoice is required by the USPS when you claim walk sequence discounts or when you are certifying.

The Delivery Sequence Invoice report is automatically generated when you include the DSF2 Walk Sequencer transform in your job.

> Note
>
> The DSF2 Walk Sequencer transform adds walk sequencing information to your data. However, to claim walk sequence postage discounts, you must further process your data using presorting software (such as the SAP BusinessObjects Presort product). There are four walk sequence discounts for which you may be eligible:
>
> - Carrier Route
> - Walk Sequence
> - 90% Residential Saturation
> - 75% Total Active Saturation

➡ Tip

The Delivery Sequence Invoice report can be large, sometimes exceeding 1000 pages. The number of pages relates to the number of postcode1/sortcode route combinations in your input.

4.9.8.1 Contents of report

The Delivery Sequence Invoice contains header information and information for each collection of data processed. Header information includes the licensee name, customer ID, site location, and processing date. The body of the report lists data for each postcode/sortcode route combination. The list below describes the data reported in the Delivery Sequence Invoice.

Data	Description
Total Deliveries	Indicates the total number of deliveries within the specified postcode/ sortcode combination.
Total Residences	Indicates the total number of residences within the specified postcode/ sortcode combination.
Delivery Points Sequenced	Indicates the number of delivery points sequenced by the transform for the specific postcode/sortcode combination
Residences Sequenced	Indicates the number of residences sequenced for the specific postcode/sortcode combination.
Percent Residences	Indicates the percent of the residences sequenced by the transform for the specified postcode/sortcode combination. The formula is 100 (residences sequenced ÷ total residences).
Percent Active Deliveries	Indicates the percent of delivery points sequenced by the transform for the specified postcode/sortcode compination.The formula is 100 (delivery points sequenced ÷ total deliveries).

Discounts	Indicates the type of discount the postcode/sortcode combination is eligible for.
	A = Sortcode Route (Carrier Route)
	B = 125 Walk Sequence
	C = 90% Residential Saturation
	D = 75% Total Active Saturation

4.9.9 US Addressing Report

The US Addressing Report provides a detailed breakdown of the various codes returned by DPV, DSF2, LACSLink, and SuiteLink processing. DPV, SuiteLink, and LACSLink are mandatory for USPS address processing according to CASS guidelines.

Related Information

Designer Guide: Data Quality, Beyond the basic address cleansing, Multiple data source statistics reporting [page 679]

4.9.9.1 To enable the report

To enable the US Addressing Report with the USA Regulatory Address Cleanse transform, complete the following options or configure the appropriate substitution variables:

- In the Report and Analysis group, select `<Yes>` for the *Generate Report Data* option.
- In the Reference Files group, set the *DPV Path*, *DSF2 Augment Path*, *LACSLink Path*, and *SuiteLink Path* as appropriate (you can use substitution variables).
- In the Assignment options group, set *Enable DPV*, *Enable DSF2 Augment*, *Enable LACSLink*, and *Enable SuiteLink* to `<Yes>` as appropriate.

4.9.9.2 Percent calculation

Many areas of the USA Regulatory Address Cleanse transform reports include percentages. The software calculates the denominator portion of the percentage calculation based on the following for the U.S. Addressing Report and the Address Validation Summary report:

US Addressing Report: The calculation of the denominator is based on the total of all DPV return codes of Y, D, S, and the addresses that are not DPV valid.

Address Validation Summary report: The calculation of the denominator is based on the total of all records processed by the USA Regulatory Address Cleanse transform.

For both reports, the following record types will no longer be factored into the percentage calculation:

- Number of records where all input fields mapped into the transform contain NULL or blank values.
- Number of records that contain Unicode characters.
- Number of records that are not processed because of Z4Change functionality.

Related Information

US Addressing Report [page 2003]
Address Validation Summary report [page 2016]

4.9.9.3 Information in the US Addressing report

The first section of the US Addressing Report includes the report title, the date of the report, and other general information about your job as follows:

- The job name
- The run ID, which specifies a unique identification of an instance of the executed job
- The name of the repository where the job is located
- The official transform name
- The path to the transform in this format: `<data flow name>` / `<transform name>`
- Audit information, which specifies how many records were processed during address cleansing, the start and end time, any idle time, and the amount of active processing time

The remainder of the report includes:

- LACSLink summary
- Counts for the LACSLink return codes encountered during processing
- DPV summary
- Counts for the DPV return codes encountered during processing
- SuiteLink return codes
- DSF2 counts from the Second Generation Delivery Sequence File (DSF2) Summary
- DSF2 Address Delivery Types

> Note
>
> Pre and post NCOALink processing sections appear in the report when NCOALink is enabled in the job. Pre NCOALink counts exist for DPV, DSF2, LACSLink, and SuiteLink. Post NCOALink counts exist for DPV and DSF2 counts/statistics.

4.9.9.4 DPV sections

The US Addressing Report contains the following sections for DPV:

Section name	Contents
Delivery Point Validation (DPV) Summary	Separate DPV number and percent for each DPV status indicator of Y, S, and D.

Section name	Contents
DPV Return Codes	Number and percent for DPV validated/not valid, CMRA validated, DPV vacant addresses, and DPV NoStats addresses.

When processing for NCOALink, the US Addressing Report includes pre and post NCOALink processing information for the Delivery Point Validation (DPV) Summary and DPV Return Codes sections.

4.9.9.5 Information in the DSF2 sections

The US Addressing Report contains the following sections for DSF2:

Section name	Contents
Second Generation Delivery Sequence File (DSF2) Summary	Number and percentage of addresses in your file that match these DSF2 address categories: • Drop addresses (including CMRA) • Business addresses • Throwback addresses • Seasonal addresses • Educational Institute addresses When you are processing for NCOALink, the US Addressing Report includes pre-NCOALink processing information and post-NCOALink processing information for this section.
Address Delivery Types	Number and percentage of addresses for these delivery type categories: • Curb • Central • Door Slot • Neighborhood Delivery Centralized Box Unit When processing for NCOALink, the US Addressing Report includes pre and post NCOALink processing information for this section.

4.9.9.6 LACSLink sections

The US Addressing Report contains the following sections for LACSLink:

Section name	Contents
Locatable Address Conversion Summary	Number and percent of records that were converted and the number and percentages of addresses that were not converted through LACSLink.
LACSLink Return Codes	Number and percent of records for each return code type.

When processing for NCOALink, the US Addressing Report includes pre and post NCOALink processing information for the Locatable Address Conversion Summary and the LACSLink Return Codes sections.

4.9.9.7 SuiteLink sections

The US Addressing Report contains the following section for SuiteLink:

Section name	Contents
SuiteLink Return Codes	The number and percent of SuiteLink matches for the following return codes: *A:* Secondary exists and assignment made. *00:* Lookup was attempted but no assignment.

When processing for NCOALink, the US Address Report includes SuiteLink Return Codes only for pre NCOALink processing.

4.9.10 DSF2 Augment Statistics Log File

The USPS requires DSF2 licensees to save monthly information about their processing in the DSF2 Augment Statistics Log File. The USPS has a specific naming format for the log files:

```
[DSF2_Licensee_ID][mm][yy].dat
```

Access the DSF2 Augment Statistics Log File through the Management Console.

> i Note
>
> The DSF2 Log File is based on when the job completes, not when the job starts. For example, a job that is started on 31 December and completes on 1 January will be included in the January DSF2 Log File.

The USPS dictates the content of the DSF2 Log File and requires that you submit log to them monthly. For details, see the DSF2 Licensee Performance Requirements document, which is available on the USPS RIBBS website. The DSF2 licensee must keep the log file data for at least 5 years.

You must submit the DSF2 Augment Statistics Log File to the USPS (National Customer Support Center) within the first seven (7) calendar days of each month.

Related Information

Exporting DSF2 certification log [page 1879]

4.9.11 US Regulatory Locking Report

Description

The software generates this report only when it encounters a false positive address during DPV or LACSLink processing with the USA Regulatory Address Cleanse transform.

The USPS includes false positive addresses with the DPV and LACSLink directories as a security precaution. Depending on what type of user you are, the behavior varies when the software encounters a false positive address.

If you use DPV or LACSLink processing for purposes other than NCOALink or if you are an NCOALink end user without an alternate stop processing agreement, the US Regulatory Locking Report contains the false positive address record (lock record) and lock code. You need this information in order to retrieve the unlock code from the SAP Service Marketplace.

If you are an NCOALink service provider or end user with an alternate stop processing agreement, the US Regulatory Locking Report contains the path to the DPV or LACSLink log files. The log files must be submitted to the USPS.

For more information about DPV and LACSLink locking and unlocking, see the *Designer Guide*.

To enable this report

To enable this report with the USA Regulatory Address Cleanse transform, verify the following options, or configure the appropriate substitution parameters:

- In the Report And Analysis options group, ensure *Generate Report Data* is set to Yes.
- In the Reference Files options group, set the *DPV Path* or *LACSLink Path* as appropriate.
- In the Assignment options group, set *Enable DPV* or *Enable LACSLink* to Yes.

How to read this report

The first section of the report includes the report title, the date of the report, and other general information about your job as follows:

- The job name
- The run ID, which specifies a unique identification of an instance of the executed job
- The name of the repository where the job is located

- The official transform name
- The path to the transform in the form data <flow name>/<transform name>
- Audit information, which specifies how many records were processed during address cleansing, the start and end times, any idle time, and the amount of active processing time

Depending on your user type, the second section of the report contains either the lock code as well as the information related to the record that caused the directory locking or the path to the DPV or LACSLink log files.

Related Information

Designer Guide: DPV security [page 619]
Designer Guide: LACSLink security [page 629]
Designer Guide: Data Quality, Beyond the basic address cleansing, USPS DPV, DPV locking, Alternate stop processing agreement [page 620]

4.9.12 Canadian SERP report: Statement of Address Accuracy

Description

The Canadian Software Evaluation and Recognition Program (SERP) Statement of Address Accuracy report includes statistical information about Canadian address cleanse processing such as the Address Accuracy Level.

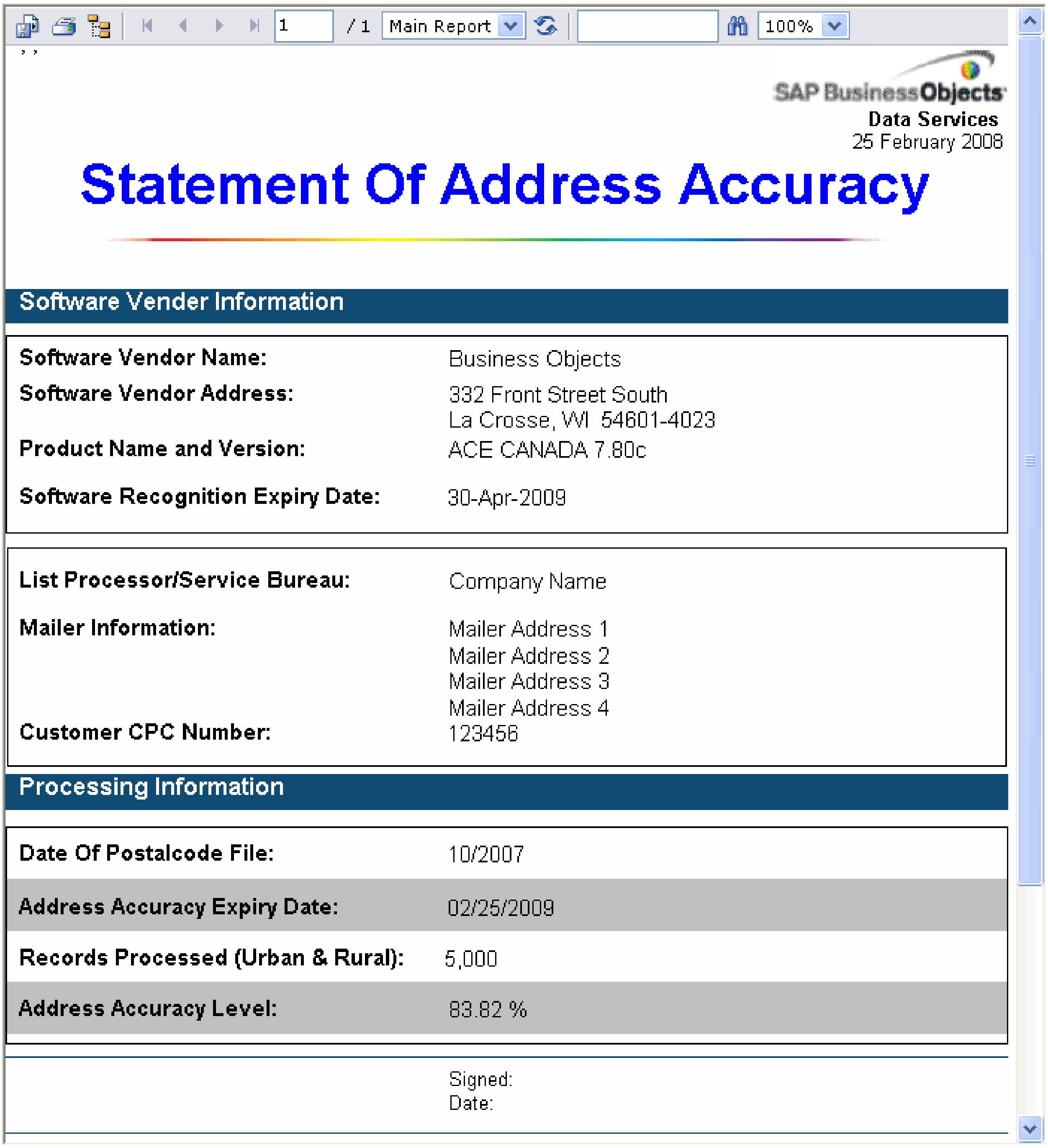
To generate the report

To generate this report with the Global Address Cleanse transform, ensure the following options are defined, or configure the appropriate substitution parameters:

- In the Global Address Cleanse transform, enable ▶ *Report And Analysis* ▸ *Generate Report Data* ◀.
- In the *Canada* group, complete all applicable options in the *Report Options* subgroup.
- In the *Engines* section, set *Canada* to *Yes*.

Example

The following is an example of the Statement of Address Accuracy report.



SAP BusinessObjects
Data Services
25 February 2008

Statement Of Address Accuracy

Software Vender Information

Software Vendor Name:	Business Objects
Software Vendor Address:	332 Front Street South La Crosse, WI 54601-4023
Product Name and Version:	ACE CANADA 7.80c
Software Recognition Expiry Date:	30-Apr-2009

List Processor/Service Bureau:	Company Name
Mailer Information:	Mailer Address 1 Mailer Address 2 Mailer Address 3 Mailer Address 4
Customer CPC Number:	123456

Processing Information

Date Of Postalcode File:	10/2007
Address Accuracy Expiry Date:	02/25/2009
Records Processed (Urban & Rural):	5,000
Address Accuracy Level:	83.82 %

Signed:
Date:

4.9.13 Australian AMAS report: Address Matching Processing Summary

Description

The Australian Address Matching Approval System (AMAS) Address Matching Processing Summary report includes statistical information about Australian address cleanse processing.

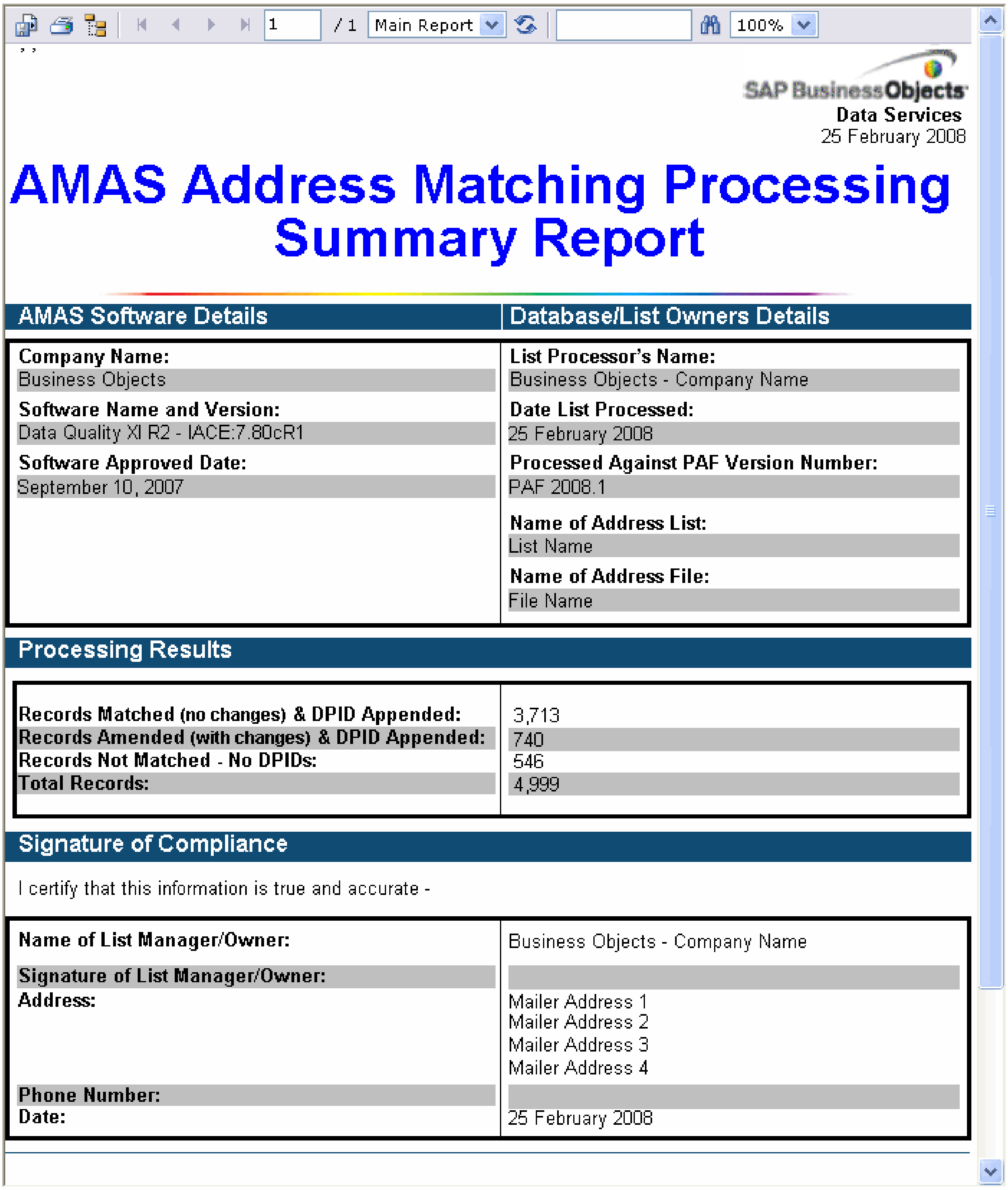
To generate the report

To generate this report with the Global Address Cleanse transform, ensure the following options have been defined, or configure the appropriate substitution parameters:

- In the Global Address Cleanse transform, enable *Report And Analysis* › *Generate Report Data*.
- In the *Australia* group, complete all applicable options in the *Report Options* subgroup.
- In the *Engines* section, set *Australia* to *Yes*.

Example

The following is an example of the Address Matching Processing Summary report.



SAP BusinessObjects
Data Services
25 February 2008

AMAS Address Matching Processing Summary Report

AMAS Software Details

Company Name:
Business Objects

Software Name and Version:
Data Quality XI R2 - IACE:7.80cR1

Software Approved Date:
September 10, 2007

Database/List Owners Details

List Processor's Name:
Business Objects - Company Name

Date List Processed:
25 February 2008

Processed Against PAF Version Number:
PAF 2008.1

Name of Address List:
List Name

Name of Address File:
File Name

Processing Results

Records Matched (no changes) & DPID Appended:	3,713
Records Amended (with changes) & DPID Appended:	740
Records Not Matched - No DPIDs:	546
Total Records:	4,999

Signature of Compliance

I certify that this information is true and accurate -

Name of List Manager/Owner:	Business Objects - Company Name
Signature of List Manager/Owner:	
Address:	Mailer Address 1 Mailer Address 2 Mailer Address 3 Mailer Address 4
Phone Number:	
Date:	25 February 2008

4.9.14 New Zealand Statement of Accuracy (SOA) report

Description

The New Zealand Statement of Accuracy (SOA) report includes statistical information about address cleansing for New Zealand.

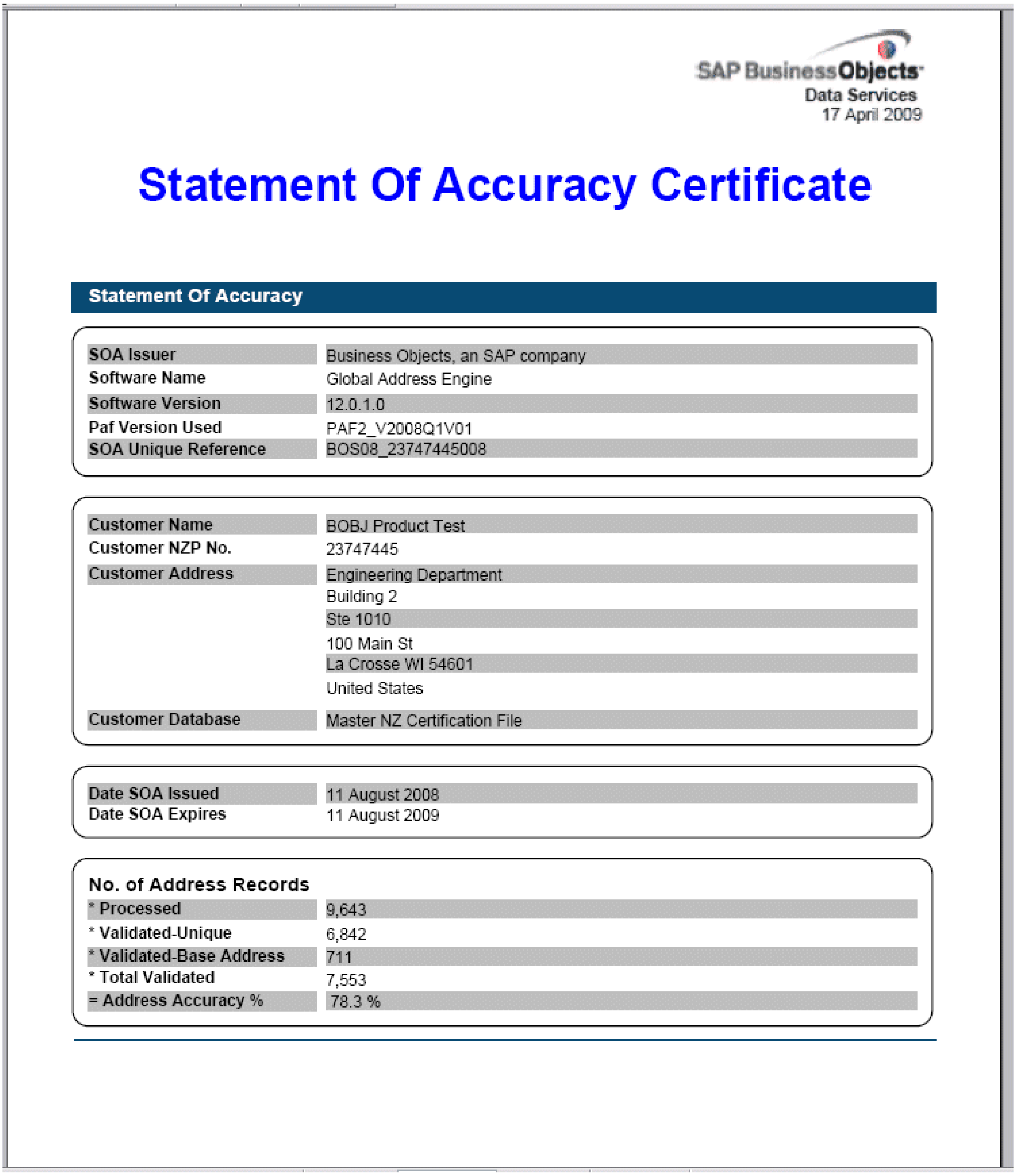
To enable the report

- In the Global Address Cleanse transform, enable ▶ *Report And Analysis* › *Generate Report Data* ◢.
- In the Global Address Cleanse transform, set ▶ *Country Options* › *Disable Certification* ◢ to *No*.
- Complete all applicable options in the ▶ *Global Address* › *Report Options* › *New Zealand subgroup* ◢.
- In the *Engines* section, set *Global Address* to *Yes*.

> **Note**
>
> The software does not produce the SOA Report when Global Address certification is disabled or when there are no New Zealand addresses included in the present job.

Example

The following is an example of the New Zealand Statement of Accuracy report.



SAP BusinessObjects
Data Services
17 April 2009

Statement Of Accuracy Certificate

Statement Of Accuracy

SOA Issuer	Business Objects, an SAP company
Software Name	Global Address Engine
Software Version	12.0.1.0
Paf Version Used	PAF2_V2008Q1V01
SOA Unique Reference	BOS08_23747445008

Customer Name	BOBJ Product Test
Customer NZP No.	23747445
Customer Address	Engineering Department Building 2 Ste 1010 100 Main St La Crosse WI 54601 United States
Customer Database	Master NZ Certification File

Date SOA Issued	11 August 2008
Date SOA Expires	11 August 2009

No. of Address Records

* Processed	9,643
* Validated-Unique	6,842
* Validated-Base Address	711
* Total Validated	7,553
= Address Accuracy %	78.3 %

Related Information

Management Console Metadata Administrator Guide: Exporting New Zealand SOA certification logs [page 1879]

Designer Guide: New Zealand SOA Report and SOA production log [page 613]

Reference Guide: Report options for New Zealand [page 1217]

4.9.15 Address Information Codes Sample report

Description

The Address Information Codes Sample report is a sampling of the records that were assigned information codes (Global Address Cleanse) or fault codes (USA Regulatory Address Cleanse) during processing. The transform uses these codes to indicate why it was unable to standardize the address. These codes can help you correct the data in specific records or find a pattern of incorrect data entry.

The software initially outputs the first fault record encountered. After that, it outputs every 100th fault record (for example 1, 101, 201, and so on). There is a maximum of 500 fault records.

> Note
>
> Depending on your configuration settings, you might see different information/fault records each time the transform processes a particular set of input records.

How to read this report

The first section of the report includes the report title, the date of the report, and other general information about your job as follows:

- Job name
- Run ID, which specifies a unique identification of an instance of the executed job
- Repository name where the job is located
- Full transform name
- Path to the transform in the form <data flow name>/<transform name>
- Engine name. For the Global Address Cleanse transform, it is the name of the global engine that processed the data, and for the USA Regulatory Address Cleanse transform, it is always USA.

The second section of the report includes a table that specifies the field details for each record for which an information/fault code was found. The table is subdivided by country.

The third section of the report lists a description of each information/fault code.

Related Information

Reference Guide: Address Cleanse reference, Information codes (Global Address Cleanse) [page 1409]
Reference Guide: Address Cleanse reference, USA Regulatory Address Cleanse transform fault codes [page 1417]

4.9.16 Address Information Code Summary report

Description

The Address Information Code Summary report provides record counts of each information or fault code of a specific project.

How to read this report

The first page of the report is a summary of all the information codes if one of the following is true:

- The job contains more than one USA Regulatory Address Cleanse or Global Address Cleanse transform.
- A single Global Address Cleanse transform processes records from more than one engine.

Subsequent pages will include a report for each transform or engine.

The first section of the summary page includes the report title, the date of the report, and other general information about your job as follows:

- Job name
- Run ID, which specifies a unique identification of an instance of the executed job
- Repository name where the job is located
- Official transform name
- Path to the transform in the form <data flow name>/<transform name>
- Audit information, which specifies how many records were processed during address cleansing, the start and end time, any idle time, and the amount of active processing time

The second part of the report includes a bar graph that shows how many different information/fault codes were assigned during processing. With this graph, you should be able to see which information/fault code occurred the most frequently, which could help you detect any consistent problems with your data.

The section below the bar graph shows how many different information/fault codes occurred along with a description of each code. For the Global Address Cleanse transform, this section is organized by engine name. At the end of the listing, the report shows the total number of information/fault codes assigned.

Related Information

Reference Guide: Transforms, Information codes (Global Address Cleanse) [page 1409]
Reference Guide: Transforms, USA Regulatory Address Cleanse transform fault codes [page 1417]

4.9.17 Address Validation Summary report

Description

The Address Validation Summary report provides record validation statistics for each Global Address Cleanse transform or USA Regulatory Address Cleanse transform of a specific job.

How to read this report

The first section includes the report title, the date of the report, and other general information about your job as follows:

- Job name
- Run ID, which specifies a unique identification of an instance of the executed job
- Repository name where the job is located
- Official transform name
- Path to the transform in the form `<data flow name>` / `<transform name>`
- Audit information, which specifies how many records were processed during address cleansing, the start and end time, any idle time, and the amount of active processing time

The second section of this report includes a bar graph that shows the output fields and how many were unchanged or corrected during address cleansing.

The third section of this report also shows the same data as in the second section, but in a table.

If NCOALink processing is enabled, this report also displays pre and post NCOALink processing graphs and statistics.

Related Information

Percent calculation [page 2003]

4.9.18 Address Type Summary report

Description

The Address Type Summary report contains record counts of each Assignment_Type field value used per Global Address Cleanse transform or Address_Type field value per USA Regulatory Address Cleanse transform of a specific job.

How to read this report

The first section of the report includes the report title, the date of the report, and other general information about your job as follows:

- Job name
- Run ID, which specifies a unique identification of an instance of the executed job
- Repository name where the job is located
- Official transform name
- Path to the transform in the form <data flow name>/<transform name>
- Audit information, which specifies how many records were processed during address cleansing, the start and end time, any idle time, and the amount of active processing time

The second section includes a pie chart that shows the percentage for the values of the Assignment_Type field (Global Address Cleanse) and the Address_Type field (USA Regulatory Address Cleanse) used in the transform.

The third section of this report also shows this same data as in the second section, but in a table.

If NCOALink processing is enabled, this report also displays pre and post NCOALink processing graphs and statistics.

Related Information

Reference Guide: Data Quality Fields, Global Address Cleanse fields [page 1224]
Reference Guide: Data Quality Fields, USA Regulatory Address Cleanse fields [page 1349]

4.9.19 Address Standardization Sample report

Description

The Address Standardization Sample report shows records where fields changed during processing. The fields displayed are your input fields and the associated output fields. Status codes are on the report to indicate why the change was necessary. This information helps you to determine which fields are frequently incorrect. You can also use this report to verify that your addresses are standardized correctly.

How to read this report

The first section of the report includes the report title, the date of the report, and other general information about your job as follows:

- Job name
- Run ID, which specifies a unique identification of an instance of the executed job

- Repository name where the job is located
- Full transform name
- Path to the transform in the form <data flow name>/<transform name>
- Engine name. For the Global Address Cleanse transform, it is the name of the global engine that processed the data, and for the USA Regulatory Address Cleanse transform, it is always USA.

The second section of the report includes the table that shows which records had fields that were standardized along with the status code that explains why the change was necessary. The section also shows the country and the input source taken from the Physical Source field.

The final page of the report has a description of the status codes.

Related Information

Reference Guide: Transforms, Status codes (Global Address Cleanse) [page 1412]
Reference Guide: Transforms, Status Codes (USA Regulatory Address Cleanse) [page 1418]

4.9.20 Address Quality Code Summary report

Description

The Address Quality Code Summary report provides record counts of each quality code assigned per Global Address Cleanse transform for a specific job. This report is not available for the USA Regulatory Address Cleanse transform.

How to read this report

The first page of the report is a summary of all the information codes if:

- The job contains more than one Global Address Cleanse transform.
- A single Global Address Cleanse transform processes records from more than one engine.

In these cases, subsequent pages will include a report for each transform or engine.

The first section of the report includes the report title, the date of the report, and other general information about your job as follows:

- Job name
- Run ID, which specifies a unique identification of an instance of the executed job
- Repository name where the job is located
- Official transform name
- Path to the transform in the form <data flow name>/<transform name>
- Audit information, which specifies how many records were processed during address cleansing, the start and end time, any idle time, and the amount of active processing time

The second section of the report includes a pie chart that shows the percentage of the quality codes that were assigned during processing. This chart illustrates the level of quality of your data.

The third section of the report shows, in a table format, the quality codes that were assigned, the record count, and a description of each quality code. This section is also divided by engine name and the country.

Related Information

Reference Guide: Transforms, Quality codes (Global Address Cleanse) [page 1416]

4.9.21 Data Cleanse Information Code Summary report

Description

The Data Cleanse Information Code Summary report provides record counts of each information code for a specific job. The information codes identify characteristics about the data. For example:

- some data went to the Extra output fields
- some person data has a family name, but not a given name
- a date's year was converted from 2-digits to 4-digits

How to read this report

The first section of the summary page includes the report title, date, and other general information about your job as follows:

- Job name
- Run ID, which specifies a unique identification of an instance of the executed job
- Repository name where the job is located
- Official transform name
- Path to the transform <dataflow name>/<transform name>
- Audit information, which specifies how many records were processed during data cleansing, the start and end time, any idle time, and the amount of active processing time.

The report begins with a summary of the frequency of information codes. The rest of the charts show the information codes per parser such as Date, Person, Firm, Phone, and so on. With the information in this report, you should be able to see which information codes occurred most frequently. This information could help you detect any consistent problems with the data.

To enable the report

In the Designer, choose ▶ *Tools* ▸ *Substitution Parameter Configurations* ◀ and configure the *Generate Report Data* option to [$$ReportsDataCleanse]. This is a parameter that can be configured to either *Yes* or *No* for report generation.

Then, in the Data Cleanse transform, enable ▶ *Report And Analysis* ▸ *Generate Report Data* ◀

Related Information

Data Services Reference Guide: Transforms, Data Quality transforms, Report and analysis [page 1149]

4.9.22 Data Cleanse Status Code Summary report

Description

The Data Cleanse Status Code Summary report provides record counts for each generated status code. The status codes describe the standards applied in the Data Cleanse transform.

How to read this report

The first section of the summary page includes the report title, date, and other general information about your job as follows:

- Job name
- Run ID, which specifies a unique identification of an instance of the executed job
- Repository name where the job is located
- Official transform name
- Path to the transform <dataflow name>/<transform name>
- Audit information, which specifies how many records were processed during data cleansing, the start and end time, any idle time, and the amount of active processing time.

The charts include a list of status code names, descriptions, and the number of records that used each status code. The report includes a chart and data for each parser, for example, Date, Phone, Firm, Person, and so on. With this report, you should have a better understanding of the generated status codes that show how the data is standardized and can be used for better matching results with the Match transform.

To enable the report

In the Designer, choose ▸ *Tools* ▸ *Substitution Parameter Configurations* and configure the *Generate Report Data* option to [$$ReportsDataCleanse]. This is a parameter that can be configured to either *Yes* or *No* for report generation.

Then, in the Data Cleanse transform, enable ▸ *Report And Analysis* ▸ *Generate Report Data*

Related Information

Data Services Reference Guide: Transforms, Data Quality transforms, Report and analysis [page 1149]

Data Services Reference Guide: Transforms, Data Quality transforms, Data Cleanse reference, Status Codes (Data Cleanse) [page 1425]

4.9.23 Geocoder Summary report

Description

The Geocoder Summary report includes statistical information about geocoding.

To enable the report

In the Geocoder transform, enable ▸ *Report And Analysis* ▸ *Generate Report Data*.

Related Information

Reference Guide: Transforms, Data Quality transforms, Report and analysis [page 1180]

4.9.24 Overview of match reports

Setting up match reports

To set up the Physical Source Field, Logical Source Field, and matching process transforms, refer to the *Designer Guide:* Data Quality, Match section. Verify the appropriate substitution parameter configurations.

Common match report information

The first section includes general information about your job such as:

- The job name
- The run ID, which specifies a unique identification of an instance of the executed job
- The repository where the job is located
- The official transform name
- The path to the transform in the form `<data flow name>` / `<transform name>`

4.9.25 Best Record Summary report

Description

The purpose of best record post-processing is to salvage data from matching records—that is, members of match groups—and consolidate, or post, that data to a best record or to all matching records. The Best Record Summary report shows statistics about the best record process used in the match. It indicates what settings were selected and the results of the posting.

If your results show any trends that could be improved by adjustments to your settings, then change those settings and re-process the step.

How to read this report

The Best Record Contribution table shows the overall results of the posting. The report columns are as follows.

Report column	Description
Best Record Name	The name of the Best Record operation you specified in the Match transform.
Posting Destination	The destination (either Master, Subordinate, or ALL) for the post.
Post Only Once Per Destination	Setting shows *Yes* or *No* to indicate whether more than one posting will be attempted for each record.
Post Attempts	The number of Best Record operations that were attempted. This is the total of the protected drops, destination field drops, filter drops, and posts completes.
Protect Drops	The number of operations that were cancelled because a posting destination was protected.
Destination Field Drops	Operations that were canceled because the Best Record operation was set to allow posting only once per destination record.
Strategy Drops	Operations that were canceled because the Best Record strategy returned *False*.
Post Completes	The number of Best Record operations that successfully completed.

The next section of the report contains audit information such as:

- How many records were processed during the match
- The start and end time
- Any idle time
- The total amount of time it took to run the Match transform

The Best Record Strategy section shows:

- The name of the best record strategy field containing the statistics
- The source and destination fields used in the best record action.

Related Information

Designer Guide: Data Quality, Set up for match reports [page 584]
Reference Guide: Transforms, Post-match processing [page 1295]

4.9.26 Match Contribution report

Description

The Match Contribution report provides you with information on the effect of the individual break groups and individual criteria on the total matching process. By studying the largest and smallest break groups and their break keys, you can determine whether they must be realigned for more accurate distribution in the matching process. The software generates one report for each match set.

You can also look at the criteria that are making the highest percentage of match/no-match decisions to verify the accuracy as well as the effectiveness of the criteria settings.

The size of the break groups has a significant impact on the speed of the matching process. If you have many large break groups (break groups with large numbers of records), the matching process slows down because it has to do so many comparisons. If that is the case, you might want to adjust your break group formation strategy so the resulting break groups are smaller.

Results showing break groups with only a single record could indicate criteria that is too restrictive. Because data in one break group is never compared with data in another break group, the restrictive criteria could be isolating data that would otherwise match.

Use this report in conjunction with the Match Criteria Summary report to understand the contributions of the various criteria and to view the detailed criteria definitions.

Break Group Contribution

The Break Group Contribution table lists the smallest and largest break group contributors. This section contains data only if you create break groups in your job.

Other information includes:

Total number of break groups	The number of break groups created based on your break group settings.
Theoretical maximum comparisons	The number of comparisons that would be made without using any break group strategy (or putting all records in a single break group).
Break group comparisons	The number of actual comparisons made because of breaking.
Comparisons per hour	The number of comparisons made per hour.
Hours saved by breaking	The amount of time saved because of breaking.
Max records in compare buffer	The maximum number of records that can fit in the memory buffer used for comparison. This value, in conjunction with the largest break group size, can be used to fine-tune performance of match process. If the largest break group size is smaller than the number of records that can fit in the compare buffer, then the records will be stored and accessed from memory. This makes the process go faster. However, if the largest break group is bigger, then some caching will be involved and it may slow down processing. In order to fix it, you can either change the breaking strategy to make smaller break groups or you can increase the buffer size by doing the following: 1. In the Designer, Choose ▸ *Tools* ▸ *Options* ▸ *Job Server* ▸ *General* ◢. 2. In the *Section* box, type **`MatchSettings`**. 3. In the *Key* box, type **`MemoryInKBForComparisons`**. 4. In the *Value* box, type the number of kilobytes of memory you want as your buffer. The default value is 4096 KB. 5. Click *OK*.

Audit Information

The Audit Information section specifies how many records were processed during the match, the start and end time, any idle time, and the total amount of time it took to run the Match transform.

The audit section is followed by the Match Set Name and the Match Level Name(s) if the match set has multiple levels.

Match Criteria Contribution

The last section shows the Match Criteria Contribution listed by criteria name and in order of the criteria execution. Focus on the criteria that are the most productive—that is, the criteria making the most decisions. The criteria that make the most decisions should be the first to be evaluated, so order your criteria appropriately in your Match transform. This will help with performance.

Related Information

Designer Guide: Data Quality, Set up for match reports [page 584]
Reference Guide: Transforms, Match transform options, Group forming [page 1269]
Match Criteria Summary report [page 2025]

4.9.27 Match Criteria Summary report

Description

The software generates one Match Criteria Summary report per match set. Use the report to obtain a consolidated view of all key settings and the criteria settings. Using this report can help you determine whether the values you set for each criteria are giving you the results you require.

After reading this report in conjunction with the Match Contribution report, which shows the break group contributions, you might decide to adjust field compare lengths or settings like match/no match score. The Match Criteria Summary report gives you the necessary information to fine-tune the settings and to compare multiple match criteria for consistency.

In the audit information section of the report, you will find information such as:

- How many records were processed during the match
- The start and end time
- Any idle time
- The total amount of time it took to run the Match transform.

The audit section is followed by the Match Set Name and the Match Level Name(s) if the match set has multiple levels.

The Match Input Fields table shows settings for the input fields used for this match including the criteria name, match criteria, field compare length, and any preprocessing options such as punctuation and casing.

The Detailed Criteria Definition table gives you a more detailed breakdown of the match criteria definitions listed by criteria in the order of execution. A list of the Match Level Options displays below the table so you can verify your settings.

> Note
>
> In cases where the *Blank Field Operation* is set to *EVAL*, the evaluation score appears as a number instead of the word IGNORE.

Related Information

Reference Guide: Transforms, Match transform options [page 1264]
Match Contribution report [page 2023]

4.9.28 Match Duplicate Sample report

Description

The Match Duplicate Sample report provides some duplicate records as a sample of the match results. One report is generated for each Match transform in the job. If a given transform results in no matches, the software does not generate a report for that transform.

The samples are taken from a maximum of 500 records per transform starting from group number 1, using every 10th match group, and up to 20 records from the selected match groups.

The Physical Source Field table displays the data sources used in the job, including the number of records sampled for that data source.

The Match Duplicate Results table displays the Match duplicate results including record number, group number, score, match type, logical source ID, and input field(s). The records are listed by group number in ascending order. If you do not see data in the Logical Source Field column, be sure that you have included an Input Sources operation in the Match transform, or defined the field that contains the logical source value.

> Note
>
> Depending on your configurations settings, you might see different information/fault records each time the transform processes a particular set of input records.

Related Information

Designer Guide: Data Quality, Set up for match reports [page 584]

4.9.29 Match Input Source Output Select report

Description

The Match Input Source Output Select report shows you which types of records and the number of records that were flagged to be kept or dropped per source.

- *Keeps:* Records you selected in the Output Flag Selection operation of the Match transform (Match Editor).
- *Drops:* Records you did not select in the Output Flag Selection operation of the Match transform (Match Editor).

Report columns

Column	Description
Net Input	The number of records in the source.
Single Source Masters	Highest ranking member of a match group whose members all came from the same source. Can be from Normal or Special sources.
Single Source Subordinate	A record that is a subordinate member of a match group whose members all came from the same source. Can be from Normal or Special sources.
Multiple Source Masters	Highest ranking member of a match group whose members came from more than one source. Can be from Normal or Special sources.
Multiple Source Subordinate	A record that is a subordinate member of a match group whose members came from more than one source. Can be from Normal or Special sources.
Suppress Masters	A record that came from a Suppress source and is the highest ranking member of a match group.
Suppress Subordinate	A record that came from a Suppress source and is a subordinate member of a match group.
Suppress Matches	Subordinate member of a match group that includes a higher-priority record that came from a Suppress source. Can be from Normal or Special type sources.
Suppress Uniques	Records that came from a Suppress source for which no matching records were found.
Uniques	Records that are not members of any match group. No matching records were found. These can be from sources with a Normal or Special source type.
Net Output	The number of records in the source minus the number of records dropped.
% Kept	The percentage of records that were not dropped from the original input source.

Related Information

Reference Guide: Match transform, Output flag selection options [page 1306]

4.9.30 Match Multi-source Frequency report

Description

The Multi-source Frequency report shows, for each input source, how many of its records were found to match records in other sources.

The format of the Multi-source Frequency report is always the same. There is a row for each of the input sources of the job. The columns show, first, the name of the source, then the total number of the source's records that appeared in more than one source. The remaining columns show how many records in that source were found in 2 sources, 3 sources, 4 sources, and so on.

If you created source groups, multi-source matches for the source groups are included in a separate table in the report.

- If a record from source1 matches a record from source2, then that record is included in the number in the 2 source column. If a record from source1 matches a record from source2 and also a record from source4, then that record is included in the number in the 3 source column.
- The entry in each column shows the number of multi-source instances—that is, how many records appeared on more than one source, not how many times they appeared. For example, if a record from source1 matches three records from source2, then that record adds one to the this source's total in the 2 source column—it's not added to the 4 source column, nor is three added to the 2 source column.
- When determining the number of sources on which a record appeared, the software does not count single-source matches, or any matches to records from special or suppression sources.

4.9.31 Match Source Statistics Summary report

Description

The Match Source Statistics report provides information about the distribution of the duplicates found in various input source records including how the duplicates were distributed as master records and subordinate records. The duplicates could be distributed within one logical source (intrasource matches) or across multiple logical sources (intersource matches). This report shows the distribution of the duplicates in general and then the results for each source. The software generates one report per match set.

To generate a meaningful Match Source Statistics report, you must have a Group Statistics operation in your Match or Associate transform. You must also generate basic statistics in those Group Statistics operations.

If you also add an Input Source operation and generate input source statistics in the Group Statistics operations, you will find additional columns related to suppression records

How to read this report

A pie chart shows what percentage of the duplicates were multiple source masters, multiple source subordinates, single source masters, and single source subordinates.

The source percentages are also detailed in a key list, indicating the distribution of the masters and subordinates.

Match Statistics table

The Match Statistics table lists the statistics for each source, enabling you to understand the impact of statistics such as total record counts and individual source counts. You can compare the number of masters and subordinates generated to the source record counts.

The columns list how many masters came from a single source or from multiple sources. Statistics are also given for the number of subordinates derived from a single source and from multiple sources.

Additional columns for input source statistics

If you chose to count input source statistics in the Group Statistics operation, you will also find the following columns.

Column	Description
Suppress Matches	The number of records that match a record in a Suppress-type source.
Total Non Dupes	The number of master records and uniques in the input source.
Suppress Uniques	The number of unique records in the Suppress-type source.
Suppress Masters	The number of master records in the Suppress-type source.
Suppress Subordinate	The number of subordinate records in the Suppress-type source.

Source by Source Statistics table

The Source by Source Statistics table details the information by source, generating statistics for the number of duplicates that are both intersource (between distinct sources) and intrasource (within a single source).

The Source Name and Comparison Source Name columns display how many times a subordinate record in the specified comparison source was found when the master was from the named source. The other two columns (Number of Inter-Source Matches and Number of Intra-Source Matches) display how many matches were found with the master in one source and the subordinate in another source (intersource) and how many matches were found with the master and subordinate in the same source (intrasource).

For example, you can compare data from two sources: Source 1 lists people who have in-ground swimming pools and Source 2 lists people who have children under the age of two. Your goal is to find the people that fit in both categories. With this report, you can now send your Swimming Pool Safety pamphlet to pool owners who have children under the age of two.

Related Information

Designer Guide: Data Quality, Set up for match reports [page 584]
Reference Guide: Transforms, Post-match processing [page 1295]
Reference Guide: Transforms, Match [page 1263]
Reference Guide: Transforms, Associate [page 1132]

5 Workbench Guide

5.1 Introduction

5.1.1 Welcome to SAP Data Services

5.1.1.1 Welcome

SAP Data Services delivers a single enterprise-class solution for data integration, data quality, data profiling, and text data processing that allows you to integrate, transform, improve, and deliver trusted data to critical business processes. It provides one development UI, metadata repository, data connectivity layer, run-time environment, and management console—enabling IT organizations to lower total cost of ownership and accelerate time to value. With SAP Data Services, IT organizations can maximize operational efficiency with a single solution to improve data quality and gain access to heterogeneous sources and applications.

5.1.1.2 Documentation set for SAP Data Services

You should become familiar with all the pieces of documentation that relate to your SAP Data Services product. The latest Data Services documentation can be found on the *SAP Help Portal*.

Document	What this document provides
Adapter SDK Guide	Information about installing, configuring, and running the Data Services Adapter SDK
Administrator Guide	Information about administrative tasks such as monitoring, lifecycle management, security, and so on.
Customer Issues Fixed	Information about customer issues fixed in this release. ℹ Note In some releases, this information is displayed the Release Notes.
Designer Guide	Information about how to use Data Services Designer.
Documentation Map	Information about available Data Services books, languages, and locations.
Installation Guide for Windows	Information about and procedures for installing Data Services in a Windows environment.
Installation Guide for UNIX	Information about and procedures for installing Data Services in a UNIX environment.
Integrator Guide	Information for third-party developers to access Data Services functionality using web services and APIs.

Document	What this document provides
Master Guide	Information about the application, its components and scenarios for planning and designing your system landscape. Information about SAP Information Steward is also provided in this guide.
Management Console Guide	Information about how to use Data Services Administrator and Data Services Metadata Reports.
Performance Optimization Guide	Information about how to improve the performance of Data Services.
Reference Guide	Detailed reference material for Data Services Designer.
Release Notes	Important information you need before installing and deploying this version of Data Services.
Technical Manuals	A compiled, searchable, “master” PDF of core Data Services books: ● *Administrator Guide* ● *Designer Guide* ● *Reference Guide* ● *Management Console Guide* ● *Performance Optimization Guide* ● *Integrator Guide* ● *Supplement for J.D. Edwards* ● *Supplement for Oracle Applications* ● *Supplement for PeopleSoft* ● *Supplement for Salesforce.com* ● *Supplement for Siebel* ● *Supplement for SAP* ● *Workbench Guide*
Text Data Processing Extraction Customization Guide	Information about building dictionaries and extraction rules to create your own extraction patterns to use with Text Data Processing transforms.
Text Data Processing Language Reference Guide	Information about the linguistic analysis and extraction processing features that the Text Data Processing component provides, as well as a reference section for each language supported.
Tutorial	A step-by-step introduction to using Data Services.
Upgrade Guide	Information to help you upgrade from previous releases of Data Services and release-specific product behavior changes from earlier versions of Data Services to the latest release.
What's New	Highlights of new key features in this SAP Data Services release. This document is not updated for support package or patch releases.
Workbench Guide	Provides users with information about how to use the Workbench to migrate data and database schema information between different database systems.

In addition, you may need to refer to several Supplemental Guides.

Document	What this document provides
Supplement for SAP	Information about interfaces between Data Services, SAP Applications, SAP Master Data Services, SAP NetWeaver BW, and SAP Master Data Services.
Supplement for SuccessFactors	Information about interfaces between Data Services and SuccessFactors.
Supplement for Salesforce.com	Information about how to install, configure, and use the SAP Data Services Salesforce.com Adapter Interface.
Supplement for J.D. Edwards	Information about interfaces between Data Services and J.D. Edwards World and J.D. Edwards OneWorld.
Supplement for Oracle Applications	Information about the interface between Data Services and Oracle Applications.
Supplement for PeopleSoft	Information about interfaces between Data Servicesand PeopleSoft.
Supplement for Siebel	Information about the interface between Data Services and Siebel.

We also include these manuals for information about SAP BusinessObjects Information platform services.

Document	What this document provides
Information platform services Administrator Guide	Information for administrators who are responsible for configuring, managing, and maintaining an Information platform services installation.
Information platform services Installation Guide for UNIX	Installation procedures for SAP BusinessObjects Information platform services on a UNIX environment.
Information platform services Installation Guide for Windows	Installation procedures for SAP BusinessObjects Information platform services on a Windows environment.

5.1.1.3 Accessing documentation

You can access the complete documentation set for SAP Data Services in several places.

5.1.1.3.1 Accessing documentation on Windows

After you install SAP Data Services, you can access the documentation from the Start menu.

1. Choose *Start* > *Programs* > *SAP Data Services 4.2* > *Data Services Documentation* > *All Guides*.
2. Click the appropriate shortcut for the document that you want to view.

5.1.1.3.2 Accessing documentation on UNIX

After you install SAP Data Services, you can access the documentation by going to the directory where the printable PDF files were installed.

1. Go to `<LINK_DIR>/doc/book/en/`.
2. Using Adobe Reader, open the PDF file of the document that you want to view.

5.1.1.3.3 Accessing documentation from the Web

You can access the complete documentation set for SAP Data Services from the SAP Business Users Support site.

To do this, go to *http://help.sap.com/bods*.

You can view the PDFs online or save them to your computer.

5.1.1.4 SAP information resources

A global network of SAP technology experts provides customer support, education, and consulting to ensure maximum information management benefit to your business.

Useful addresses at a glance:

Address	Content
Customer Support, Consulting, and Education services *http://service.sap.com/*	Information about SAP support programs, as well as links to technical articles, downloads, and online forums. Consulting services can provide you with information about how SAP can help maximize your information management investment. Education services can provide information about training options and modules. From traditional classroom learning to targeted e-learning seminars, SAP can offer a training package to suit your learning needs and preferred learning style.
Product documentation *http://help.sap.com/bods/*	SAP product documentation.
Supported Platforms (Product Availability Matrix) *https://service.sap.com/PAM*	Get information about supported platforms for SAP Data Services. Use the search function to search for Data Services. Click the link for the version of Data Services you are searching for.
SAP Data Services Community Network *http://scn.sap.com/community/data-services*	Get online and timely information about SAP Data Services, including forums, tips and tricks, additional downloads, samples, and much more. All content is to and from the community, so feel free to join in and contact us if you have a submission.
Blueprints *http://scn.sap.com/docs/DOC-8820*	Blueprints for you to download and modify to fit your needs. Each blueprint contains the necessary SAP Data Services project, jobs, data flows, file formats, sample data, template

Address	Content
	tables, and custom functions to run the data flows in your environment with only a few modifications.
SAPTerm *https://portal.wdf.sap.corp/go/sapterm*	SAP's terminology database, the central repository for defining and standardizing the use of specialist terms.

5.1.2 Overview of this guide

Welcome to the *Workbench Guide*. The Data Services Workbench provides a graphical user interface (GUI) development environment in which you define data application logic to migrate data and database schema information between different databases in a data warehousing environment.

Key features of the Workbench include:

- Browsing table metadata and data.
- Selecting individual source tables for migration.
- Specifying the order in which the source tables should be migrated.
- Adjusting the table schema in detail. For example, adding or removing columns, defining constraints, partitions, indexes, and so on.
- Specifying filters and simple projection expressions.
- Specifying source and target table options such as array fetch size and bulk-loading options.
- Executing a replication job as an initial load or delta load.
- Modeling and consuming a data quality job.

5.1.2.1 About this guide

This guide contains two kinds of information:

- Conceptual information that helps you understand the Data Services Workbench and how it works.
- Procedural information that explains in a step-by-step manner how to accomplish a task.

You will find this guide most useful:

- While you are learning about the product.
- While you are performing tasks in the design and early testing phases of your data-movement projects.
- As a general source of information during any phase of your projects.

5.1.2.2 Who should read this guide

This and other Data Services product documentation assumes the following:

- You are an application developer, consultant, or database administrator working on data warehousing.

- You understand your source data systems and RDMBS.
- You understand your organization's data needs.
- You are familiar with SQL (Structured Query Language).
- You are familiar with Data Services installation environments — Microsoft Windows or UNIX.

5.1.3 Naming Conventions

In this documentation, the following naming conventions apply:

Terminology

- “Data Services system” refers to “SAP Data Services”.
- “BI platform” refers to “SAP BusinessObjects BI platform”.

  > Note
  >
  > The BI platform components required by Data Services may also be provided by SAP BusinessObjects Information platform services (IPS).

- “CMC” refers to the Central Management Console provided by the BI or IPS platform.
- “CMS” refers to the Central Management Server provided by BI or IPS platform.

Variables

Variables	Description
`<INSTALL_DIR>`	The installation directory for the SAP software. Default location: • `C:\Program Files (x86)\SAP BusinessObjects` • `$HOME/sap businessobjects`
`<BIP_INSTALL_DIR>`	The root directory of the BI or IPS platform. Default location: • **`<INSTALL_DIR>`**`\SAP BusinessObjects Enterprise XI 4.0` • **`<INSTALL_DIR>`**`/enterprise_xi40` Note These paths are the same for both the SAP BusinessObjects BI platform and SAP BusinessObjects Information platform services.

Variables	Description
`<LINK_DIR>`	The root directory of the Data Services system. Default location: - All platforms **`<INSTALL_DIR>`**`/Data Services` This system environment variable is created automatically during installation.

`<DS_COMMON_DIR>`	The common configuration directory for the Data Services system.
	Default location:
	- Windows (Vista and newer)
	`ALLUSERSPROFILE\SAP BusinessObjects\Data Services`
	- Windows (Older versions)
	`ALLUSERSPROFILE\Application Data\SAP BusinessObjects\Data Services`
	- UNIX systems (for compatibility)
	`<LINK_DIR>`
	This system environment variable is created automatically during installation.

`<DS_USER_DIR>`	The user-specific configuration directory for the Data Services system.
	Default location:
	- Windows (Vista and newer)
	`USERPROFILE\AppData\Local\SAP BusinessObjects\Data Services`
	- Windows (Older versions)
	`USERPROFILE\Local Settings\Application Data\SAP BusinessObjects\Data Services`
	This user environment variable is created automatically during installation.
	i Note
	This variable is used only for Data Services client applications on Windows, such as the Designer. **`<DS_USER_DIR>`** is not used on UNIX platforms.

5.2 Workbench User Interface

This section provides basic information about the Workbench user interface.

5.2.1 Logging into the Workbench

Before you can use the Workbench to create, edit, and execute replication jobs, you must log in as a user defined in the Central Management Server (CMS).

1. Launch the Workbench.

   The Workbench can be accessed from the Windows Start Menu: *All Programs* › *SAP Data Services* **<x.x>** › *Data Services Workbench*.
2. Enter your user credentials for the CMS.

Option	Description
System	The server name and optionally the port number for the CMS.
User name	The user name to use to log into the CMS.
Password	The password to use to log into the CMS.
Authentication mode	The authentication type used by the CMS.

3. Click *Connect*.
   The software attempts to connect to the CMS using the specified information. If the authentication is successful, the Workbench window is displayed.

5.2.2 Workbench window

By default, the Workbench user interface consists of several primary elements:

- Menu bar
- Toolbar
- Project explorer
  The Workbench project explorer contains the current project, and all jobs and datastores that are a part of it.
- Workspace
  The workspace is the area of the Workbench window where you define, display, and modify objects. When you create a new object, the Workbench automatically opens a new editor tab in the workspace. The workspace is also used to display the data of a source table and to display the job status and execution logs.

Workspace editors

The workspace area of the Workbench contains tabs for the editors you currently have open. The editors that may appear in the workspace include the following:

- Replication job editor
  The replication job editor allows you to display and edit replication job options. For example, the target table schema, projections expressions and filters, and replication groups. The name of the open replication job is indicated in the tab's title. If the replication job contains unsaved data, it is indicated with an asterisk (*).
- Data Flow editor
  The data flow editor allows you to create data flows and include them in the replication job. The supported tranforms are Query transform, Case transform, Merge transform, Row Generation transform, MAP Operation transform, and SQL transform. You can reference existing data flows from the Replication Job by dragging and dropping the data flows in from the Explorer. You can verify all data flows and specify whether the data flow is for initial load, delta load, or both.
- Datastore editor

The datastore editor allows you to display and edit datastore properties, browse and import metadata, and display and edit table properties. The name of the open datastore is indicated in the tab's title. If the datastore contains unsaved data, it is indicated with an asterisk (*).

- File Format editor
  The file format editor allows you to set properties for file format templates and source and target file formats.
- Monitoring editor
  The monitoring editor displays your replication jobs' status, execution progress, and any applicable logs. Trace, monitor, and error logs are available.

Related Information

Replication job editor [page 2041]
Datastore editor [page 2045]
Data Flow editor [page 2043]
Monitoring editor [page 2057]
File Format editor [page 2046]

5.2.3 Menu bar

This section contains a brief description of the Workbench menus.

5.2.3.1 File menu

The File menu contains standard Windows as well as software-specific options.

Option	Description
New	Define a new folder, project, datastore, data flow, or replication job.
Open File	Open an existing Workbench file.
Close	Close the active Workbench editor.
Close All	Close all open editors.
Save	Save the object in the active editor.
Save All	Save all objects in open editors.
Exit	Exit the Workbench.

5.2.3.2 Edit menu

The Edit menu contains standard Windows options.

Option	Description
Undo	Undo the last operation.
Redo	Redo the last undone operation.
Cut	Cut the selected objects or text and place it on the clipboard.
Copy	Copy the selected objects or text to the clipboard.
Paste	Paste the contents of the clipboard into the active editor or text box.
Delete	Delete the selected objects or text.
Select All	Select all objects in the active editor or all text in the active text box.

5.2.3.3 Tools menu

The Tools menu contains replication job editor and datastore editor options.

Option	Description
Open monitoring editor	Open the job monitoring editor.
Quick replication wizard	Start the quick replication wizard.
Export to ATL	Export a selected replication job to the local file system.
Deploy	Deploy a replication job or datastore definition to a Data Services repository.
Execute	Deploy a replication job to a Data Services repository and execute the job. Note This option is available only for the replication job editor.

5.2.3.4 Validate menu

The Validate menu contains replication job editor and datastore editor validation options.

Option	Description
Validate	Validate the active Workbench replication job for errors.

Option	Description
	ℹ Note The *Validate* option is available only for the replication job editor.
Show DDL	Display the data definition language (DDL) for the active Workbench object. ℹ Note The *Show DDL* option is available only for the replication job editor and only applies when the replication job editor is active.
Show ATL	Display the ATL for the active Workbench object. ℹ Note Do not use this option to create a file for import into Data Services. Use the ▶ *Tools* ▶ *Export Replication Job option* ◢ to export a replication job.

5.2.3.5 Window menu

The Window menu provides software-specific window options.

Option	Description
Data Cleansing Solutions	Display the Data Cleansing Solutions tab in the workspace area.
Error Log	Display the Error Log tab in the workspace area.
Problems	Display the Problems tab in the workspace area.
Project Explorer	Display the project explorer.
Reference Analysis	Display a reference impact analysis for the active Workbench object in the workspace area.
Other	Display other window panes.
Reset Perspective	Reset the Workbench interface perspective to its default settings.
Preferences	Display the Preferences window.

5.2.3.6 Help menu

The Help menu provides standard help options.

Option	Description
Welcome	Display the Workbench welcome screen.
Help	Display the SAP Data Services Workbench Guide.
About	Display information about the software including the version of the Workbench and copyright information.
Key Assist	Display information about keyboard shortcuts available in the Workbench.

5.2.3.7 Replication job editor

The Workbench replication job editor opens in the workspace when you create a new replication job or open an existing replication job from the project explorer.

The replication job editor consists of several areas:

- Project explorer
  Displays the name of the source datastore and all imported tables.
- Target datastore pane
  Displays the name of the target datastore and all mapped tables and columns. In this pane, mapped objects may be grouped into replication groups.
- Replication job configuration pane
  When you select an object such as a datastore table or column in the replication job editor, its configuration options are displayed in this pane.
  The option groups that are available depend on the type of object that is selected:

Object	Available options
Project explorer	Properties
Replication job	Properties Variables Scripts
Replication group	Properties
Target table	Properties Columns Foreign Keys Indexes Options Filter data DDL Options

Object	Available options
	Delta load
Column	Properties

You can adjust the size of each pane by dragging the borders between them. In addition, you can hide or reveal the replication job configuration pane by clicking the arrows in the center of its border.

Related Information

Using the replication job editor [page 2068]

5.2.4 Toolbar

In addition to many of the standard Windows tools, the software provides application specific tools, including:

Icon	Tool	Description
🗋	New	Define a new folder, project, datastore, or replication job.
💾	Save	Save the object in the active editor.
✂	Cut	Cut the selected objects or text and place it on the clipboard.
	Copy	Copy the selected objects or text to the clipboard.
📋	Paste	Paste the contents of the clipboard into the active editor or text box.
✕	Delete	Delete the selected objects or text.
	Open monitoring editor	Open the job monitoring editor.
✓	Validate	Validate the active Workbench resource for errors.
DDL	Show DDL	Display the data definition language (DDL) for the active Workbench resource.
ATL	Deploy ATL	Deploy the ATL for the active replication job to the Data Services repository.
▶	Execute	Deploy the ATL for the active replication job to the Data Services repository, and execute the replication job.
	Quick replication wizard	Start the quick replication wizard to easily define source and target datastores and create a basic replication job.

5.2.5 Project explorer

The project explorer contains the projects currently defined in the user workspace. Each Workbench project is represented as a folder containing all datastores and replication jobs associated with that project.

When you create new projects, datastores, data flows, file formats, and replication jobs, they are displayed in the project explorer. You can also use the project explorer to import and export Workbench project information.

5.2.6 Data Flow editor

Workbench Data Flow editor enables you to create data flows and include them in the replication job. The supported tranforms are the Query transform, Case transform, Merge transform, Row Generation transform, MAP Operation transform, and SQL transform. You can reference existing data flows from the Replication Job by dragging and dropping the data flows in from the Explorer. You can verify all data flows and specify whether the data flow is for initial load, delta load, or both.

You also have the ability to convert replication tables to data flows. Highlight multiple replication tables, right click the selection, and select convert data. You have the option to generate a single data flow or one data flow per replicated table.

Data Flow editor options

The Workbench Data Flow editor includes the Data Flow diagram, a list of input schemas for the selected transform, a list of available Transforms, and a tabbed area containing the properties of the object currently selected in the Data Flow diagram.

Object	Available options
Data Flow diagram	The Data Flow diagram is composed of a graphical area and a palette of elements you can drop on the graphical area. The Data Flow diagram toolbar includes options, such as Zoom slider, Snap to Grid, Highlight Related Element, and so on.
Data Flow diagram toolbar	The Data Flow diagram toolbar includes the following: • Zoom slider • Snap to grid • Enable highlight related elements • Enable descriptions in the diagram • Summary mode • Save a screenshot of the diagram
Transforms	This panel displays all available transforms. Drag and drop the transforms onto the Data Flow diagram.
Input Schemas panel	This panel displays all the available input schemas / fields that the user can drag-and-drop to the properties area. Every field that is being used in the current prop-

Object	Available options
	erty tab is marked with a black triangle. Some fields are marked with a key icon indicating they are part of the primary key.
Object properties	Displays the properties of the object that is currently selected in the Data Flow diagram.

The options for Table Reader properties, File Reader properties, Table Loader properties, Merge properties, Case properties, SQL properties, Row Generation properties, Map Operation properties, Date Generation properties, and the Effective Date properties are the same as the Data Services Designer.

The Template Table Loader uses the same database specific loader options as Data Services Designer as well as some extra options:

- Use NVARCHAR for VARCHAR
- Drop and re-create table
- Double quote names
- Table Type (for HANA only)

Use data file as source or target

You can select and drop data from the Project Explorer onto the Data Flow diagram and choose to use it as a source or target.

Validation in the Data Flow editor

Objects containing errors will be underlined in red and a tooltip will show the list of errors/warnings for an element.

5.2.6.1 Additional Data Flow options

The Workbench Data flow Editor includes several options that are not available in Designer Data Flow editor.

Template file loader

Located in the Transforms panel, the Template File loader enables you to add data to a file without having to create a file format. Drop the Template file loader onto the Data Flow diagram and set the template file properties. You must enter the fully qualified file name(s).

Detect Mappings

Located on the Mappings tab, the Detect mappings option assists you during target to source mapping by detecting source fields with similar names. The dialog provides a list of proposed candidate mappings for you to view and confirm. Every proposed candidate field has a matching score (from 0 to 100), and proposals are displayed from the highest score to the lowest to allow you to see the most relevant proposals first. Only the proposals that are confirmed are applied.

Detect Joins

Join detection combines both foreign key relationships analysis (if the source has foreign keys) and name, data type matching. Every join that is detected is assigned a detection score and shown in a confirmation dialog. A join found using foreign key detection will always have a score of 100, and a detection using name will have a score between 0 and 99 depending on the similarity of the fields that are proposed as a join.

Create Query transform

The Data Flow feature allows you to right-click and select to add a new query transform to the data flow. Workbench will create the query transform object, connect it to both selected sources, connect it to the loader, run the detect joins, and detect mapping.

The Data Flow detections page under Preference gives you the flexibility to be more restrictive for the minimum score and allows you to choose whether to always confirm the detections or not.

Related Information

Data Flow Detections [page 2057]

5.2.7 Datastore editor

The Workbench datastore editor opens in the workspace when you create a new datastore or open an existing datastore from the project explorer or replication job editor.

The datastore editor consists of these areas:

- Navigation pane
  The navigation pane displays the configuration and imported tables and views of the datastore. For SAP datastores, the navigation pane also displays the imported SAP extractors.
  When the datastore configuration is active, you can edit the datastore properties and settings. When the datastore tables, extractors, or views are active, you can import or re-import tables, extractors, and views, view data, and compare imported metadata with the metadata in the database.

- Object properties pane
  Displays tabs for the properties and options for the object currently selected in the navigation pane. The option tabs that are available depend on the type of object that is selected:

Object	Available option
Table	Properties Columns Foreign Keys Indexes Partitions Attributes
Column	Properties Attributes

You can adjust the size of each pane by dragging the borders between them. In addition, you can hide or reveal the navigation pane by clicking the arrow in the center of its border.

Related Information

Using the Datastore editor [page 2067]

5.2.8 File Format editor

A file format describes the structure of an ASCII file. A file format consists of multiple properties. You set the properties in the *File Format editor*.

The work area on the left, in the *File Format editor*, lists file format properties that are not field specific. The following table lists all of the options. These options are filtered by the mode you are using.

Option	Possible values	Description	Mode
General			
File format	Delimited	The format of the data in the text file. Available properties change based on the selected file format type.	New, Edit
Adaptable file reading	Yes, No	Indicates whether the schema of a delimited file format is adaptable or fixed. • *Yes* indicates that the schema is adaptable. The actual file can contain fewer or more columns than indicated by the file format. If a row contains fewer columns than expected, the software loads null val-	New, Edit, Source

Option	Possible values	Description	Mode
		ues into the columns missing data. If a row contains more columns than expected, the software ignores the additional data. • *No* indicates that the schema is fixed. The software requires the number of columns in each row to match the number of columns specified in the file format. The default is *No*. If you select *Yes*, you must ensure that the selected column and row delimiters do not appear inside the actual data.	
Custom transfer	Yes, No	Enables the software to use a third-party file transfer program. Displays additional Custom transfer program options in the *File Format Editor* below the Input/ Output properties.	All
Skip error handling	Yes, No	Selecting *Yes* disables the Error Handling section in the Format editor. The default is *No.*	New, Edit, Source
Number of processing threads	Integer greater than 0, {none}, {default}	Specifies the number of threads for parallel processing. Parallel processing can improve performance by maximizing CPU usage on the Job Server computer. For example, if you have four CPUs, enter 4 for this option. For more information, see “File multi-threading” in the *Performance Optimization Guide*. For jobs that process USPS certification tests, the value should be set to *{none}*.	All
Data File(s)			
File location	Local, Job Server	During design, indicates the files are located on the local machine or on the machine that runs the Job Server. If you select Job Server, you must enter the absolute path to files. Remember that UNIX systems are case-sensitive. During execution, all files must be located on the Job Server machine that executes the job. If you use different files to design your job, change the file specified (through the Root directory and File properties) before you execute the job.	All
Root folder	Path name for the file or blank	The directory where the file is located. For added flexibility, you can enter a variable for this option. If you enter a directory name, then enter only the file name for the *File* property.	New, Edit

Option	Possible values	Description	Mode
		If you leave the root directory blank, then enter a file name that includes the full path name in the *File* property.	
File name(s)	File name(s), file name(s) including full path name, or blank	In new and edit modes, specifies an existing file on which you base the file format description. Data from this file appears in the Column Attributes area. In these modes, you can leave this property blank. In source and target modes, specify the location of the actual file for this source or target. In these modes, you cannot leave this property blank. For added flexibility, you can enter: ● A variable that is set to a particular file with full path name. Use variables to specify file names that you cannot otherwise enter, such as file names that contain multi-byte characters. ● A list of files, separated by commas, or a file name containing a wildcard. In this case, the software reads all these files as a single source. See “Reading multiple files at one time” in the *Designer Guide*.	All
Delimiters			
Column delimiter	Tab, Semicolon, Comma, Space, or any character sequence	For delimited file formats, the character sequence that indicates the end of one column and the beginning of the next.	New, Edit
Row delimiter	{new line}, {Windows new line}, {Unix new line}, or any character sequence	A character sequence that indicates where one row of data ends and the next begins.	New, Edit
Row delimiter within text string	Character, Row delimiter	Defines how the row delimiter is interpreted within a text string. ● *Character*: The specified row delimiter is treated as a character within the text string. ● *Row delimiter*: The specified row delimiter is interpreted and defines rows within the text string.	New, Edit
Text delimiter	Single quotation marks ('), double quotation marks ("), or {none}	Denotes the start and end of a text string. All characters (including those specified as column delimiters) between the first and second occurrence of this character is a single text string. The treatment of the row characters is defined by the *Row within text string* setting.	New, Edit

Option	Possible values	Description	Mode
ℹ Note Data in columns cannot include the column or row delimiter, unless you also specify a text delimiter. For example, if you specify a comma as your column delimiter, none of the data in the file can contain commas. However, if you specify a comma as the column delimiter and a single quote as the text delimiter, commas are allowed in strings in the data. You can use any ASCII characters (including non-printing characters) for column and row delimiters. You can specify an ASCII character by entering a forward slash (/) followed by the decimal representation of the character. For example, to use Y umlaut (Ÿ) as a delimiter, enter /255 in the delimiter property box.			
Default Format			
Escape character	Any character sequence, or {none}	A special character sequence that causes the software to ignore the normal column delimiter. Characters following the escape character sequence are never used as column delimiters. For example, suppose you specify a forward slash as the escape character and a comma as the column delimiter. Then, you must have a forward slash to have a comma appear inside a field.	New, Edit
NULL indicator	{none} or any other character sequence	A special character sequence that the software interprets as NULL data. The software ignores any NULL indicator specified in the file format for blob columns.	New, Edit
Ignore row marker(s)	{none} or any other character sequence	A character sequence, which when found at the beginning of rows, causes the software to ignore the row when reading the file or automatically creating metadata. To enter multiple character sequences, separate each with a semi-colon. To include a semi-colon or backslash as a marking character, precede with a backslash.	New, Edit, Source
Date format	yyyy.mm.dd or other combinations	The date format for reading or writing date values to and from the file.	New, Edit
Time format	hh24:mi:ss or other combinations	The time format for reading or writing time values to and from the file.	New, Edit
Date-Time format	yyyy.mm.dd hh24:mi:ss or other combinations	The datetime format for reading or writing datetime values to and from the file.	New, Edit

Option	Possible values	Description	Mode
Input/Output			
Style	Headers or BOF/EOF	The format of the start and end of the file. Available properties in the Input/Output property group may change, based on this selection.	New, Edit
Skipped rows	Integer	For file formats using Headers style, the number of rows skipped when reading the file. Specify a non-zero value when the file includes comments or other non-data information.	New, Edit
Skip row header	Yes, No	For file formats using the Headers style, indicates whether the first row of data in the file contains the column names and should be skipped when reading the file. The software uses this property in addition to the *Skipped rows* property. When you select *Yes*, the software does not read data from the first row, and uses data in the first row to determine the file's column names.	New, Edit
Write row header	Yes, No	For file formats using Headers style, indicates whether to write column names in the first row of the output file.	New, Edit
Write BOM	Yes, No	For file formats using UTF-8 and UTF-16, determines the writing of BOM characters into the file. Choose Yes if you want to include BOM characters into a UTF-8 or UTF-16 file in which byte order is not otherwise defined. For a UTF-16 file, the software assumes the file to be UTF-16be, unless BOM characters are added by this property.	New, Edit
Beginning string in a file	Any character sequence, including a blank space, an empty string, or {none}	For file formats using BOF/EOF style, the string that marks the start of data in the file.	New, Edit
Ending string in a file	Any character sequence, including a blank space, an empty string, or {none}	For file formats using BOF/EOF style, the string that marks the end of data in the file.	New, Edit
Custom Transfer options			
Program executable	File name	(Required) The name of the custom transfer program or its initialization script. For example: MyProgram.exe or MyProgram.cmd.	New, Edit
User name	Any character sequence, including a blank	(Optional) Log in ID for the server to which the custom transfer program connects. You may want to allow a	New, Edit

Option	Possible values	Description	Mode
	space, an empty string, or {none}	custom program user to enter their user name when they enter their password in the software.	
Password	Any character sequence, including a blank space, an empty string, or {none}	(optional) Password for the server to which the custom transfer program connects. Passwords entered into this option are encrypted.	New, Edit
Arguments	Any character sequence, including a blank space, an empty string, or {none)	(Optional) You can create arguments in your custom transfer program and then specifically flag them from within the software using this box. For example, you might have security or compression mechanisms in your program. You can also link connection data to your transfer program's flags.	New, Edit
Locale			
Language	The three-letter language abbreviations specified in the ISO 639-2/T standard	Specifies the human language (for example, Korean, Japanese, or English) in which data is stored or processed. Select from the displayed list.	New, Edit
Code Page	The list of supported code pages. See Supported Locales and Encodings.	Specifies the sequence of bits that defines a character. For example, the Japanese code page contains ASCII, Greek, Cyrillic, and Japanese characters, thereby supporting the English, Greek, Russian, and Japanese languages.	New, Edit
Error Handling			
Log data conversion warnings	Yes, No	Determines whether to include data-type conversion warnings in the error log. The default is *Yes*.	New, Edit, Source
Log row format warnings	Yes, No	Determines whether to include row-format warnings in the error log. The default is *Yes*.	New, Edit, Source
Maximum warnings to log	Integer greater than 0 or {no limit}	If *Log data conversion warnings* or *Log row format warnings* is enabled, you can limit how many warnings the software logs. The default is *{no limit}*.	New, Edit, Source
Capture data conversion errors	Yes, No	Determines whether to capture data-type conversion errors when processing a flat-file source. The default is *No*.	New, Edit, Source
Capture row format errors	Yes, No	Determines whether to capture row-format errors when processing a flat-file source. The default is *Yes*.	New, Edit, Source
Accepted bad records in file	Integer greater than 0 or {no limit}	Specify the number of bad records in a file that will be accepted.	

Option	Possible values	Description	Mode
Write error rows to file	Yes, No	Determines whether to write invalid rows to an error file. The default is *No*.	New, Edit, Source
Error file root directory	Directory path or blank	If *Write error rows to file* is enabled, type the root directory in which to save the error file. For added flexibility, you can enter a variable for this option. If you type a directory path here, then only enter the file name in the *Error file name* property. If you leave *Error file root directory* blank, then type a full path and file name in the *Error file name* property.	New, Edit, Source
Error file name	File name, file name including full path name, or blank	If *Write error rows to file* is enabled, type the name of the file in which to record the invalid rows. For added flexibility, you can enter a variable that is set to a particular file with full path name. Use variables to specify file names that you cannot otherwise enter such as file names that contain multi-byte characters.	New, Edit, Source

The Fields work area in the *File Format editor* contains properties about the fields in the file format.

Property	Possible values	Description
Field name	Any sequence of letters or numbers, not including blank spaces	A name that identifies data in this column. If your file format uses the Headers style and you select *Yes* for the *Write row header* property, the software writes the field names in the target file.
Data type	blob, date, datetime, decimal, double, integer, interval, long, numeric, real, time, varchar	The data type of values in this column. The long data type is not available in fixed-width formats.
Format	{none}	For all data types other than varchar, specifies the format for this particular field. You can use this property to overwrite the default format. For example, if one date field is different than others, you can specify the different format here.
Nullable	Yes, No	Indicates whether the the field will accept NULL values.
Description	Any sequence of letters or numbers, not including blank spaces	Text that that specifies the type of data in a column. Typically use the field name, or a name similar to the field name. For example, if your field name is LastName, you may want to name the content type Family_Name.

If you delete a file format template from the object library, you must also delete all file sources and targets that are based on that file format template.

Related Information

Using the File Format editor [page 2076]

5.2.9 Target file properties

The following properties are available on the Target file properties tab in the *Replication Job editor*.

Option	Possible values	Description
General		
File format	Delimited	The format of the data in the text file. Available properties change based on the selected file format type.
Adaptable Schema	Yes, No	Indicates whether the schema of a delimited file format is adaptable or fixed. ● *Yes* indicates that the schema is adaptable. The actual file can contain fewer or more columns than indicated by the file format. If a row contains fewer columns than expected, the software loads null values into the columns missing data. If a row contains more columns than expected, the software ignores the additional data. ● *No* indicates that the schema is fixed. The software requires the number of columns in each row to match the number of columns specified in the file format. The default is *No*. If you select *Yes*, you must ensure that the selected column and row delimiters do not appear inside the actual data.
Custom transfer program	Yes, No	Enables the software to use a third-party file transfer program. Displays additional Custom transfer program options in the *File Format Editor* below the Input/Output properties.
Parallel process threads	Integer greater than 0, {none}, {default}	Specifies the number of threads for parallel processing. Parallel processing can improve performance by maximizing CPU usage on the Job Server computer. For example, if you have four CPUs, enter 4 for this option. For more information, see “File multi-threading” in the *Performance Optimization Guide*. For jobs that process USPS certification tests, the value should be set to *{none}*.
Data File		
Location	Local, Job Server	During design, indicates the files are located on the local machine or on the machine that runs the Job Server. If you select Job Server, you must enter the absolute path to files. Remember that UNIX systems are case-sensitive.

Option	Possible values	Description
		During execution, all files must be located on the Job Server machine that executes the job. If you use different files to design your job, change the file specified (through the Root directory and File properties) before you execute the job.
Root directory	Path name for the file	The parent directory for the specified file name. If the directory name is entered, the file will be generated there and the file name format will be: `<jobname_tablename>.<extension>`. If the directory is blank, the file will be generated under **`<DS_COMMON_DIR>`**`/workspace/`.
File Extension	File extension	The file extension to be applied to all loader files in the replication job.
Delimiters		
Column	Tab, Semicolon, Comma, Space, or any character sequence	For delimited file formats, the character sequence that indicates the end of one column and the beginning of the next.
Row	{new line}, {Windows new line}, {Unix new line}, or any character sequence	A character sequence that indicates where one row of data ends and the next begins.
Row within text string	Character, Row delimiter	Defines how the row delimiter is interpreted within a text string. - *Character*: The specified row delimiter is treated as a character within the text string. - *Row delimiter*: The specified row delimiter is interpreted and defines rows within the text string.
Text	Single quotation marks ('), double quotation marks ("), or {none}	Denotes the start and end of a text string. All characters (including those specified as column delimiters) between the first and second occurrence of this character is a single text string. The treatment of the row characters is defined by the *Row within text string* setting.
Note Data in columns cannot include the column or row delimiter, unless you also specify a text delimiter. For example, if you specify a comma as your column delimiter, none of the data in the file can contain commas. However, if you specify a comma as the column delimiter and a single quote as the text delimiter, commas are allowed in strings in the data. You can use any ASCII characters (including non-printing characters) for column and row delimiters. You can specify an ASCII character by entering a forward slash (/) followed by the decimal representation of the character. For example, to use Y umlaut (Ÿ) as a delimiter, enter /255 in the delimiter property box.		
Default Format		

Option	Possible values	Description
Escape character	Any character sequence, or {none}	A special character sequence that causes the software to ignore the normal column delimiter. Characters following the escape character sequence are never used as column delimiters. For example, suppose you specify a forward slash as the escape character and a comma as the column delimiter. Then, you must have a forward slash to have a comma appear inside a field.
NULL indicator	{none} or any other character sequence	A special character sequence that the software interprets as NULL data. The software ignores any NULL indicator specified in the file format for blob columns.
Ignore row marker(s)	{none} or any other character sequence	A character sequence, which when found at the beginning of rows, causes the software to ignore the row when reading the file or automatically creating metadata. To enter multiple character sequences, separate each with a semi-colon. To include a semi-colon or backslash as a marking character, precede with a backslash.
Date format	yyyy.mm.dd or other combinations	The date format for reading or writing date values to and from the file.
Time format	hh24:mi:ss or other combinations	The time format for reading or writing time values to and from the file.
Date-Time format	yyyy.mm.dd hh24:mi:ss or other combinations	The datetime format for reading or writing datetime values to and from the file.
Validate decimal data	Yes, No	Perform decimal validation and formatting on output.
Input/Output		
Style	Headers or BOF/EOF	The format of the start and end of the file. Available properties in the Input/Output property group may change, based on this selection.
Skipped rows	Integer	For file formats using Headers style, the number of rows skipped when reading the file. Specify a non-zero value when the file includes comments or other non-data information.
Write row header	Yes, No	For file formats using Headers style, indicates whether to write column names in the first row of the output file.
Write BOM	Yes, No	For file formats using UTF-8 and UTF-16, determines the writing of BOM characters into the file. Choose Yes if you want to include BOM characters into a UTF-8 or UTF-16 file in which byte order is

Option	Possible values	Description
		not otherwise defined. For a UTF-16 file, the software assumes the file to be UTF-16be, unless BOM characters are added by this property.
BOF marker	Any character sequence, including a blank space, an empty string, or {none}	For file formats using BOF/EOF style, the string that marks the start of data in the file.
EOF marker	Any character sequence, including a blank space, an empty string, or {none}	For file formats using BOF/EOF style, the string that marks the end of data in the file.
Custom Transfer options		
Custom transfer program executable	File name	(Required) The name of the custom transfer program or its initialization script. For example: MyProgram.exe or MyProgram.cmd.
Custom transfer user name	Any character sequence, including a blank space, an empty string, or {none}	(Optional) Log in ID for the server to which the custom transfer program connects. You may want to allow a custom program user to enter their user name when they enter their password in the software.
Custom transfer password	Any character sequence, including a blank space, an empty string, or {none}	(optional) Password for the server to which the custom transfer program connects. Passwords entered into this option are encrypted.
Custom transfer arguments	Any character sequence, including a blank space, an empty string, or {none)	(Optional) You can create arguments in your custom transfer program and then specifically flag them from within the software using this box. For example, you might have security or compression mechanisms in your program. You can also link connection data to your transfer program's flags.
Locale		
Language	The three-letter language abbreviations specified in the ISO 639-2/T standard	Specifies the human language (for example, Korean, Japanese, or English) in which data is stored or processed. Select from the displayed list.
Code Page	The list of supported code pages. See Supported Locales and Encodings.	Specifies the sequence of bits that defines a character. For example, the Japanese code page contains ASCII, Greek, Cyrillic, and Japanese characters, thereby supporting the English, Greek, Russian, and Japanese languages.

5.2.10 Monitoring editor

The Workbench Monitoring editor opens in the workspace when you click the icon in the toolbar or select it from the *Tools* menu.

The monitoring editor consists of several areas:

Area	Description
Job selection toolbar	Displays the Data Services jobs available in the repository to which the Workbench is connected and allows you to execute them. i Note The monitoring editor displays all standard Data Services jobs in the repository, not Workbench-specific replication jobs.
Execution history pane	Displays the execution history for the selected job. Each line indicates the current status, execution date, and duration of a single job execution instance.
Execution history dashboard pane	Displays a graphical representation of the history for the selected job. You can change the type of chart by choosing from the drop-down box in the upper-right corner of the pane.
Execution details pane	Displays the details for the individual job execution instance selected in the execution history pane. You can view the trace, monitor, and error logs for the execution instance, as well as the execution statistics.

You can adjust the size of each pane by dragging the borders between them. In addition, you can hide or reveal the dashboard and details panes by clicking the arrows in the center of their borders.

5.2.11 Preferences

The Preferences options allow you to Customize several areas of Data Services Workbench.

5.2.11.1 Data Services

From the main Data Services page, specify the Maximum number of errors to show in tooltips.

5.2.11.1.1 Data Flow Detections

The Data Flow Detection options allow you to customize the way detection algorithms work, for example, setting the minimum scores to display in the Detect Mapping and Detect Join dialogs.

Mappings detection

Option	Description
Minimum score threshold	Select the minimum mapping score to display. Scores at or above this setting are displayed in the Mapping detection dialog.
Enable mapping detection in create Query context menu	Check to enable mapping detection when creating a query transform from the context menu of a multiple selection of source elements of the data flow and of one Loader.
Minimum score threshold	Select the minimum mapping score to display. Scores at or above this setting are displayed in the Mapping detection dialog. This option only applies to the Enable Mapping Detection option in the create query context menu.
Always confirm mappings	Select to always confirm mappings when creating a new Query transform. This option applies to the Enable Mapping Detection option in the create query context menu.

Joins detection

Option	Description
Minimum score threshold	Select the minimum join score to display. Scores at or above this setting are displayed in the join detection dialog.
Enable join detection in create Query context menu	Check to enable mapping detection when creating a query transform from the context menu of a multiple selection of source elements of the data flow and of one loader.
Minimum score threshold	Select the minimum mapping score to display. Scores at or above this setting are displayed in the Mapping detection dialog.
Always confirm joins	Select to always confirm joins when creating a new query transform. This option applies to the Enable Join detection option in the create query context menu.

5.2.11.1.2 Data Flow editor

Data Flow preferences allow you to customize various settings of the Data Flow editor, including the way that the Auto-arrange option arranges the elements in the Data Flow diagram (for example, minimizing spacing between elements).

Option	Description
Top Margin	Specify the top margin spacing.
Left Margin	Specify the left margin spacing.
Horizontal Margin	Specify the horizontal margin spacing.
Vertical Margin	Specify the vertical margin spacing.
Horizontal Alignment	Specify the horizontal alignment: • Left Aligned • Center Aligned • Right Aligned
Automatically put names in edit mode when an element is dropped in the diagram	Check this option place the field names in edit mode when they are dropped into the diagram.

5.2.11.1.3 Data preview

Data preview preferences allow you to customize the way Data preview opens and to change the maximum number of rows and kilobytes to retrieve.

Option	Description
Max Rows to retrieve from Tables	Specify the maximum number of rows to retrieve from the database.
Max Size (in KB) to retrieve from Files	Specify the maximum number of kilobytes to retrieve from the files.
Data Preview opening mode	Specify the viewing method: • In an Editor • In a View • In a Dialog

5.2.11.1.4 Datastore editor

The Datastore editor preferences page allows you to specify whether to import the row count of the table when importing the metadata and select the table and column information you want to display.

Option	Description
Automatically detect row count when importing tables (might be slower)	Specify whether to import the row count of the table when importing the metadata. If selected, the row count will appear in the table properties and the project explorer.
Tables:	
Prefix tables with owner	Check to prefix tables with owner.
Show column count	Check to show column count.
Show row count	Check to show row count.
Show row size	Check to show row size.
Show description	Check to show the table description.
Columns:	
Show datatype	Check to show the column datatype.
Show description	Check to show the column description.

5.2.11.1.5 Expression editor

Customize the Expression editor by setting the Content assist and Syntax coloring options.

Content assist options

Option	Description
Insert single proposals automatically	Select this option to automatically insert single proposals.
Insert schema qualified column names	Select this option to schema qualified column names.
Enable auto activation	Select this option to enable automatic activation.
Activation delay (ms)	Enter the activation delay in milliseconds. Default value is 200.
Activation triggers	Select the activation triggers you want to include: '.' to propose qualified names '$' to propose variables '[' to propose substitution parameters

Syntax coloring

Use the syntax coloring options to set the color and fonts of the expression editor syntax.

5.2.11.1.6 Expression macros

These options allow you to manage the expression macros (creation, edition, and deletion) and import and export macros from this screen.

Expression macro listing

Button/option	Description
New folder	Create a new folder.
New macro	Create a new expression macro.
Delete	Delete selected expression macro(s).
Move up	Move selected macro up.
Move down	Move selected macro down.

General tab

Option	Description
Description	Describes the use for selected function.
Expression	Enter the expression macro you want to add.

Conditions tab

Option	Description
Restrict to following datatypes	Check to only apply the expression macro to select datatypes. Select the datatypes from the dropdown list.
Restrict to following mapping types	Check to apply the expression macro to select mapping types. Select mapping types from the dropdown list.

Option	Description
Restrict to the following content types	Check to only apply expression macro to select content types. Select the content types from the dropdown list.
Restrict to the following field name pattern	Check to restrict a specific field name pattern. Select the field name pattern from the dropdown list.

Other options/buttons

Option/button	Description
Import	Import the expression macros from file.
Export	Export the expression macros to a file.
Restore defaults	Restores the expression macros back to the defaults included with Data Services Workbench.

5.2.11.1.7 File Format editor

The File Format editor options allow you to specify the number of kilobytes to scan during Data Preview and the detection algorithms of files.

Option	Description
Maximum file size to use for the detect fields	Specify the number of kilobytes to scan during the detection algorithms of files.
Maximum file size to use for Data Preview	Specify the number of kilobytes to scan during data preview.

5.2.11.1.7.1 Detection settings

The Data Type detection options affect the data types detected when creating a file format from an existing file and detecting data types. If a field's values match one of the listed formats, Workbench will assign the corresponding data type to the field: Date, Time or DateTime. Workbench attempts to detect the formats in the order they appear in the list for each format.

Option	Descriptioin
DateTime Formats	Add, edit, delete, or reorder DateTime formats.
Date Formats	Add, edit, delete, or reorder Date formats.
Time Formats	Add, edit, delete, or reorder Time formats.

Option	Descriptioin
Locales	Add, edit, delete, or reorder Locales. The configured locales can affect the detection of decimal data types if thousands-separators or decimal-separators are used. For example, 10.237.329,95 will not be recognized as a decimal (10,2) unless a locale that is configured with appropriate decimal characters (German) is included in the list.
Null Values	Add, edit, delete, or reorder null values.

5.2.11.1.8 Project explorer

These options allow you to fine-tune the way Data Services Workbench project files are displayed in the project explorer. For example, you can specify how to group the tables of the Datastore file.

Table 178: General

Option	Description
Show file extensions	Check to show file extensions.

Table 179: Data stores

Option	Description
Show table count	Check to show the table count.
Show database count	Check to show the database type.
Show description	Check to show the datastore description.

Table 180: Tables

Option	Description
Show columns count	Check to show the number of columns.
Show table type	Check to show the table type.
Show row count	Check to show the table row count.
Show description	Check to show the description.
show row size	Check to show the row size.

Table 181: Tree structure

Option	Description
Group table by type	Check to group tables by type.
Group table by owner	Check to group tables by owner.
Enable table chunking	Check to enable table chunking.

Option	Description
Chunk size	If *Enable table chunking* is selected, enter the number of tables.

Table 182: Columns

Option	Description
Show data type	Check to show the data type.
Show description	Check to show the description.

5.2.11.2 General

The Preferences section allows you to set your preferences for the following:

- General
- Appearance
- Content Types
- Editors
- Keys
- Perspectives
- Workspace

5.2.11.3 Languages

Enter the product languages you want to use in Workbench. To apply language changes you must restart Workbench.

5.3 Database Migration

Migrating data and schema information between different database systems is a complex task that can take days or even months depending on the type of database systems, the complexity of the database schema, and the size of the data set being migrated. In addition, incompatibilities between the source and target database types could require manual schema and data corrections.

The Data Services Workbench simplifies this database migration process considerably. In the Workbench, you provide connection information for the source and target databases and select the tables that you want to migrate. The Workbench automatically creates Data Services jobs, work flows, and data flows and imports them into a Data Services repository.

After you have created the replication jobs, you can use the Workbench to execute the jobs and monitor their progress. If required, you can also edit the generated data flows in the Designer, and use other Data Services tools to schedule and execute the jobs.

5.3.1 Migrating data and schema information

The process of migrating data and schema information from one database to another with the Workbench generally consists of a few basic steps:

1. Create a project in the project explorer.
2. Create a source datastore definition.
3. Create a target datastore definition.
4. Create a replication job using your source and target datastores.
   a. Select the source objects that you want to migrate.
   b. Group the target objects and specify the order of migration.
   c. Adjust the target schema as required.
   d. Specify the replication behavior for the job.
5. Deploy and execute the replication job.

You can choose to perform the datastore and replication job definition steps manually with the standard individual datastore and replication job editors, or you can complete the process using the quick replication wizard. If you use the replication wizard, you can modify your replication job by using the standard editors after the wizard completes.

Related Information

Project explorer [page 2043]
Datastore editor [page 2045]
Replication job editor [page 2041]

5.3.1.1 Creating a Workbench project

If you want to create a new project, start the new project wizard in one of the following ways:

- Right-click inside the project explorer and choose *New* > *Project*.
- Choose *New* > *Project* from the *File* menu.
- Click the *New* toolbar icon. Choose *General* > *Project* and click *Next*.

1. Specify the name for the new project and click *Next*.

   > **Note**
   >
   > The project name can contain only alphanumeric characters and the underscore character.

2. If you want to save the files for the project to a different location, uncheck *Use default location* and specify a location for the project files.
3. Select the Data Services repository.
4. Click *Finish* to create the project.

The new project is displayed in the project explorer.

You can now add Workbench objects such as datastores and replication jobs to your project.

5.3.1.2 Using the quick replication wizard

You can use the quick replication wizard to easily migrate source data to one of the supported target databases or files.

1. Click the *Quick replication wizard* icon in the toolbar.
   The quick replication wizard is displayed.
2. Specify the name for the new replication project and click *Next*.

   > **i** Note
   >
   > The project name can contain only alphanumeric characters and the underscore character.

   The source datastore configuration screen is displayed.
3. Specify the settings for the source datastore and click *Next*.
   a) Select the datastore type.
   The available configuration settings for the selected database or application type are displayed.
   b) Set the datastore configuration settings as required for your database or application.

   > **i** Note
   >
   > Required settings are marked with an asterisk (*), and required settings that are missing a value are highlighted in red in the *Property name* column. Settings that have changed from the default value are listed in bold.

   c) Click *Test connection* to verify that the connection settings are valid.

   If the connection test fails, correct the datastore connection settings and re-test the connection.

   > **i** Note
   >
   > The connection test is between the database or application and the Central Management Console (CMC) host, not the Workbench host. Ensure that the connection settings between the CMC and datastore database or application are valid.

   The source object selection screen is displayed.
4. Select the objects that you want to replicate to the target datastore or files and click *Next*.
   The configure target settings screen is displayed.
5. Specify the settings for the target datastore or files and click *Next*.
   a) Select a target type of *Files* or *Datastore*
   b) If the target type is Datastore, select the datastore type.
   The available configuration settings for the selected database or application type are displayed.
   c) Set the configuration settings as required for your database, files, or application.

> **Note**
>
> Required settings are marked with an asterisk (*), and required settings that are missing a value are highlighted in red in the *Property name* column. Settings that have changed from the default value are listed in bold.

   d) When you select datastore as the target type, cick *Test connection* to verify that the connection settings are valid.

      If the connection test fails, correct the datastore connection settings and re-test the connection.

   The repository and job server selection screen is displayed.

6. If you want to execute the replication job without making any further changes, select *Execute replication job now*.
   a) In the left column, select the Data Services repository to use.

      The jobs generated by the Workbench will be stored in this repository. By default, the first repository in the list is selected.

   b) In the right column, select the Data Services Job Server to use to execute the replication job.

      By default, the Job Server associated with the selected repository is used.
7. Click *Finish* to save the replication job and and execute it with the selected settings, if applicable.
   If you chose to execute the replication job, the monitoring editor is displayed.

The Workbench saves the replication job and datastore definitions to the local file system. If you chose to execute the replication job now, it also creates Data Services jobs and datastores for the replication job, imports them to the selected repository, and executes them using the selected Job Server.

5.3.1.3 Using the Datastore editor

The datastore editor provides a powerful interface that allows you to fine-tune your datastore definition, including importing database tables, inserting or modifying columns, keys, indexes, and partitions, modifying attributes, and so on.

> **Tip**
>
> The datastores defined and used in the Workbench are standard Data Services datastores. For more information about Data Services datastores, see the *Reference Guide*.

If you have already created a datastore, open it in the datastore editor by double-clicking the datastore name in the project explorer.

If you want to create a new datastore definition, start the new datastore wizard in one of the following ways:

- Right-click your project in the project explorer and choose *New* > *Datastore*.
- Select your project in the project explorer and choose *New* > *Datastore* from the *File* menu.
- Select your project in the project explorer and click the *New* toolbar icon. Choose *Data Services* > *Datastore* and click *Next*.

1. Specify the name for the new datastore and click *Next*.

> Note
>
> The datastore name can contain only alphanumeric characters and the underscore character.

2. Specify the settings for the datastore and click *Next*.
   a) Select the datastore type.
   The available configuration settings for the selected database or application type are displayed.
   b) Set the datastore configuration settings as required for your database or application.

   > Note
   >
   > Required settings are marked with an asterisk (*), and required settings that are missing a value are highlighted in red in the *Property name* column. Settings that have changed from the default value are listed in bold.

   c) Click *Test connection* to verify that the connection settings are valid.
   If the connection test fails, correct the datastore connection settings and re-test the connection.

   The object selection screen is displayed.
3. Select the objects that you want to import to the datastore definition and click *Finish*.

The new datastore is opened in the datastore editor.

You can now use the datastore editor to further customize your datastore definition. For example, you can select a table or view and modify its properties, columns, keys, and so on, or use the configurations section to add additional configurations to your datastore.

Related Information

Datastore editor [page 2045]
Supported data migration sources and targets [page 2086]
Reference Guide: Objects, Descriptions of objects, Datastore [page 860]
Datastore editor [page 2059]

5.3.1.4 Using the replication job editor

The replication job editor provides a powerful interface that allows you to fine-tune your replication job, including filtering datastore tables, customizing target replication groups, modifying table and column mappings, and so on.

> Note
>
> The replication jobs defined and used in the Workbench are defined in Workbench-specific metadata. When you deploy the replication job to a Data Services repository, the Workbench uses this metadata to create standard Data Services objects such as job, work flows, and data flows. For more information about Data Services objects, see the *Reference Guide*.

If you have already created a replication job, open it in the replication job editor by double-clicking the replication job name in the project explorer.

If you want to create a new replication job, start the new replication job wizard in one of the following ways:

- Right-click your project in the project explorer and choose *New* > *Replication Job*.
- Select your project in the project explorer and choose *New* > *Replication Job* from the *File* menu.
- Select your project in the project explorer and click the *New* toolbar icon. Choose *Data Services* > *Replication job* and click *Next*.

1. Specify the name and description for the new replication job and click *Finish*.

   > Note
   >
   > The replication job name can contain only alphanumeric characters and the underscore character.

   The new replication job is opened in the replication job editor.
2. Select a Replication Target Type:
   - *Datastore*
   - *Files*
3. Do one of the following:
   - If you selected *Datastore*, Click *...* to specify the source and target datastores.

   A list of available datastores is displayed. If you have not already created the datastores, you can define the datastores later.

   - If you selected *Files*, set the target file properties. You must enter the root directory.

You can now use the replication job editor to further customize your replication job definition. For example, you can change the replication behavior, modify the target schema, customize column mappings, and so on.

Related Information

Replication job editor [page 2041]

Reference Guide: Objects [page 832]

5.3.1.4.1 Modifying replication job properties

When you open a replication job or select the root replication job node in the target datastore pane, the configuration area displays several tabs that you can use to configure the way that the Workbench processes the replication job:

Tab	Description
Properties	Replication target type of Datastore: Displays the replication target type, name, owner, replication behavior and description for the replication job. Replication target type of Files: Displays the replication target type, name, description and target file properties.

Tab	Description
Variables	Displays information about the variables available for use in the replication job. You can add or delete variables, as well as alter variable properties such as data type, length, precision, scale, and default value.
Scripts	Displays the pre-load and post-load scripts for the replication job. You can modify the scripts directly in the text box or by clicking *Edit* to launch the Workbench expression editor. You can also validate the scripts.

Related Information

Using the expression editor [page 2087]

5.3.1.4.1.1 Replication behavior

When you deploy and execute a Workbench replication job, the Workbench automatically generates the ETL metadata for the job and deploys it to a Data Services repository. The metadata that is generated depends on the replication behavior setting for the replication job.

The following replication behavior options are available:

- Schema and data
  In the schema and data scenario, the Workbench assumes that the tables do not exist in the target database and need to be created. As a result, the ETL job metadata that is generated includes the following objects:
  - Source and target datastores
  - Source and target tables
  - A Data Services job that contains both the script to create the target tables, as well as the work flows and data flows that move data from the source tables to the target tables.
- Data only
  In the data-only scenario, the Workbench assumes that the tables already exist in the target database. As a result, the ETL job metadata that is generated includes the following objects:
  - Source and target datastores
  - Source and target tables
  - A Data Services job that contains work flows and data flows that move data from the source tables to the target tables.

Note

Schema and data is the default replication behavior.

5.3.1.4.2 Adjusting the target schema

5.3.1.4.2.1 Modifying target table properties

When you select a table in the target datastore pane, the configuration area displays several tabs that you can use to configure the way that the Workbench processes that table:

Tab	Description
Properties	Displays the table name, owner, and description and provides the ability to view the data in the source table.
Columns	Displays information about the target table columns. After the table has been imported, you can insert and delete columns from the target table. You can also change column names, data types, descriptions, and so on, as well as alter the column mappings with the expression editor.
Foreign Keys	Displays information about the table's primary and foreign keys, if any. After the keys have been imported, you can add, delete, and modify them.
Indexes	Displays information about the table's indexes. The primary index is listed first and followed by any secondary indexes. You can add or remove indexes, as well as add or remove columns from existing indexes.
Options	Displays options that Data Services uses to read the source table and load the target table. For more information about the available source and target options, see the *Reference Guide*. i Note Only options used by the Workbench are displayed.
Filter data	Allows you to filter the rows of the source table. You can use basic filter editor or the expression editor to modify the filter.
DDL Options	Allows you to specify the behavior of DDL generation. For example, whether to create foreign keys or indexes.
Delta load	Displays options that modify the delta load behavior for a table. For more information about the available delta load options, see *Delta load migration* [page 2078].

Related Information

Reference Guide: Objects, Descriptions of objects, Source [page 947]
Reference Guide: Objects, Descriptions of objects, Target [page 960]

5.3.1.4.2.2 Grouping target tables

Many data warehouse tables have referential integrity constraints such as primary key and foreign key relationships. The primary key table needs to be loaded before the foreign key table if there is a relationship between two tables.

You can avoid referential integrity issues by using the replication job editor to group tables. For example, you can put the primary key tables in the first group and foreign key tables in the second group. In another scenario, you could put dimension tables in the first group and fact tables in the second group.

By default, tables are added to the Default_Group that is created in a new replication job. However, there is no limit to the number of replication groups supported in a Workbench replication job.

To group tables in the replication job editor:

1. Create a new replication group in the target schema.
   a) Right-click on the replication job root node in the target schema pane and choose *New replication group*.
   A new replication group is added to the target schema.
   b) Assign a meaningful name to the replication group by highlighting the new replication group and clicking it again.
2. Assign tables to the new group by dragging them from the source pane to the replication group.

   > ℹ Note
   >
   > A table may belong to only one group in the target schema. You can create multiple target tables from a single source table, but each target table must be named uniquely.

3. Use the up and down arrows in the target schema to order the replication groups as needed for your requirements. When the replication job is executed, the groups will be migrated starting from the top of the list.

   You can also rearrange replication groups by dragging them within the target schema.

   > ➡ Tip
   >
   > To automatically order the tables in the target schema by their dependencies, click *Auto order*.

> ℹ Note
>
> Replication groups can be sequential or parallel. In a sequential group, the data flows generated by the Workbench will be executed in sequential order. In a parallel group, the data flows are executed in parallel. To change the group type, right-click on the group and choose *Parallel* or *Sequential* from the *Change replication group type* menu.

5.3.1.4.2.3 Changing column data types

Data Services supports conversion to and from its internal data types and performs data type conversions when it imports metadata from external sources or targets into the repository, as well as when it loads data into an external table. When the Workbench creates target tables, the software converts from internal data types to the data types used by the target database.

For more information about internal data type conversions in Data Services, see the *Reference Guide*.

The Workbench allows you to overwrite the default data conversion rules. To change the data type for a column:

1. Expand the table containing the column in the target schema pane.
2. Select the target column.
   The column details are displayed in the configuration pane.
3. Select the new data type for the column from the *Data type* drop-down box.

   If required for the new data type, set the *Length*, *Precision*, and *Scale* for the column.

Related Information

Reference Guide: Data Types, Data type conversion, Conversion to or from internal data types [page 1040]

5.3.1.4.3 Filtering datastore tables

In a replication job, you might want to migrate only a subset of the rows from a particular source table to your target datastore.

To migrate a subset of table rows, use a filter on the datastore table in the replication job editor:

1. Select the table that you want to filter in the target schema pane.
2. Click the *Filter data* tab in the configuration pane.
3. For basic filtering, use the table filters configuration window.
   a) Select *Basic* and click *Edit filter*.
      The table filters configuration window is displayed.
   b) Drag the columns you want to use to filter from the columns pane to the filters pane.
      The columns are displayed in the filters panel and grouped together with a logical `AND` or `OR` grouping operator.
   c) For each column, define the filter logic by specifying the comparison operator and value.

      > ➡ Tip
      >
      > You can create more complex, nested filter expressions by dragging the columns on top of each other. The nesting layout is displayed in a graphical preview.

   d) Change the grouping logic as required by double-clicking the grouping operator.
   e) To remove a filter, click the column or grouping operator to highlight the filter and click the *Remove selected filter* button.

f) When you are satisfied with the filter, click *OK*.

The filter is displayed in the *Filter data* tab.

4. For more advanced filtering, use the Workbench expression editor to define the filter.
   a) Select *Advanced* and click *Edit filter*.
   The expression editor is displayed.
   b) Complete your filtering expression and click *OK*.
   The filter expression is displayed in the *Filter data* tab.
   c) To validate your filter expression, click *Validate*.

Related Information

Using the expression editor [page 2087]

5.3.1.5 Using the Data Flow editor

The Data Flow editor enables you to create data flows and include them in the replication job. You also have the ability to convert replication tables to data flows. Highlight multiple replication tables, right-click the selection, and select convert data. You have the option to generate a single data flow or one data flow per replicated table.

If you have already created a data flow, open it in the Data Flow editor by clicking the data flow name in the project explorer.

To create a new data flow, do one of the following:

- Select your project in the project explorer and right-click your data flow in the project explorer and choose ▸ *New* ▸ *Data flow* ◂.
- Select your project in the project explorer and choose ▸ *New* ▸ *Data Flow* ◂ from the *File* menu.
- Select your project in the project explorer and click the *New* toolbar icon. Choose ▸ *Data Services* ▸ *Data Flow* ◂ and click *Next*.

1. Specify the name for the new data flow and click *Next*.

   > Note
   >
   > The data flow name can contain only alphanumeric characters and the underscore character.

2. Enter the description for the new data flow and click *Finish*.

The new data flow is opened in the Data Flow editor.

Related Information

Data Flow editor [page 2043]

Expression macros [page 2075]

Reference Guide: Objects, Descriptions of objects, Data Flow [page 857]
Data Flow editor [page 2059]

5.3.1.6 Converting replication tables to data flows

To convert a replication table to a data flow:

1. Highlight the replication tables you want to convert.
2. Right-click the selection.
3. Select one of the following:
   - ○ *Generate a single data flow*
   - ○ *One data flow per replicated table*

5.3.1.7 Expression macros

You have the option to apply expression macros to multiple fields at one time when working in the Query transform of a data flow. You can access the Workbench Preferences options from the Query transform. This enables you to create new expression macros and apply them to the fields as needed.

5.3.1.7.1 Creating an expression macro

To create a new expression macro, access the Preferences Expression macros options one of the following ways:

- From the Data Flow Mappings tab, right-click a field and choose ▶ *Apply expression macro* ▶ *Manage expression macros* ◀.
- Choose *Preferences* from the *Window* menu and select the *Expression Macros* option.

1. Click the *New macro* button and enter the macro name.
2. From the *General tab*, enter a Description for the macro.
3. From the General tab, enter the Expression.
4. From the Conditions tab, select any conditions you want to apply to the expression macro.

Related Information

Expression macros [page 2061]

5.3.1.7.2 Applying an expression macro to multiple fields

To apply an expression macro to multiple fields at one time:

1. Select the fields you want to include.
2. Right-click the highlighted fields and select *Apply expression macros*.
3. Select one of the expressions from the list or select *Manage expression macros* to add a new expression macro.

5.3.1.8 Using the File Format editor

The File Format editor allows you to set properties for file format templates and source and target file formats.

If you have already created a file format, open it in the File Format editor by clicking the file format name in the project explorer.

If you want to create a new file format, start the File Format wizard in one of the following ways:

- Select your project in the project explorer, right-click and choose ▶ *New* ▸ *Flat File Format* ◀.
- Select your project in the project explorer and click the *New* toolbar icon. Choose ▶ *Data Services* ▸ *Flat File Format* ◀ and click *Next*.

5.3.1.8.1 Manually create a new file format

To manually create a file format with default properties:

1. Select *Manually create a new File Format* and click *Next*:
2. Specify the name and description for the new file format and click *Finish*.

> ℹ Note
>
> The name can contain only alphanumeric and underscore characters.

The new file format is opened in the File Format editor.

Now you can further define the properties for the file format templates using the the File Format editor.

5.3.1.8.2 Automatically import file formats

This option enables you to import multiple files with different formats at the same time and generate multiple file formats. You have the option to auto-detect or specify the file formats.

To automatically import file formats from existing files:

1. Select *Automatically import File Formats from existing files* and click *Next*:
2. Under *File location*, select Local or Job Server and click *Add files*.
3. Select the file(s) you want to add and click *Next*.
4. Do one of the following:
   - Click *Next* to set the detection options.
   - Click *Finish* to run auto-detection.
5. If you chose to set detection options, set the detection options and click *Finish*.

The new file format is opened in the File Format editor.

Now you can use the File Format editor to further define the properties for the file format templates.

5.3.1.9 Deploying and executing the replication job

When you are satisfied with your replication job configuration, you can deploy it to a Data Services repository and execute it.

1. Double-click the replication job in the project explorer to open it in the replication job editor.
2. Click *Validate* to verify that the replication job is free of errors.
3. If you want to deploy the job but not execute it, click *Deploy*.
   a) Select the repository where you want to deploy the replication job and click *OK*.
   A progress bar indicating the deployment progress is displayed.

   A message indicating the success or failure of the deployment is displayed. If any errors occur during deployment, the message includes any details available.
4. If you want to deploy and execute the job immediately, click *Execute*.
   a) Select the repository where you want to deploy the replication job and click *Next*.
   The job execution parameters screen is displayed.
   b) Specify whether to execute the job as an initial load or delta load.
   c) Specify the job server and other execution options to use when executing the job in the *Execution options* tab.
   d) If you want to include additional information in the trace log, change the values in the *Trace* tab to *Yes* as required.
   e) Click *Finish* to deploy and execute the job.
   A progress bar indicating the deployment progress is displayed. If you want to halt the deployment, click the *Cancel operation* button next to the progress bar.

   The monitoring editor for the job is displayed.

Related Information

Monitoring editor [page 2057]

5.3.1.9.1 Error recovery

If an error occurs while your replication job is being executed, only some rows may be inserted into a target table.

There are several methods that you can use to prevent duplicate rows from being inserted into the target table, including:

- Data flow design
  You can design the replication job to completely replace the target table each time the job is executed. You might choose to use this method if the number of rows changed in the target table is a significant percentage of the target table. You can also use techniques such as bulk loading options to improve the overall performance of the replication job.
- Auto-correct load target table option
  You can use the auto-correct load option to check the target table for existing rows before new rows are added to the table. However, this option can needlessly slow jobs that are not executed in recovery mode. You might choose to use this method if the number of rows changed in the target table is a relatively small percentage of the target table.
- Pre-load SQL commands
  You can use pre-load SQL commands to remove partial database updates that have occurred during an incomplete job execution. Typically, the pre-load SQL command deletes rows based on a variable set before the partial insertion began.

5.3.2 Delta load migration

Delta load jobs move rows that have been added or modified since the last time the job was executed. There are many reasons to implement a delta load job, but the most common is to reduce the time the loading process takes. Instead of loading millions of rows each time the job is run, you can process only the few that have changed. Another reason could be to maintain historical data; you might want to keep the old data in your data warehouse and add the current state so that you can see the changes over time.

5.3.2.1 Delta load jobs in the Workbench

When you create a replication job that has a table configured with a delta load method, the Workbench automatically creates two sets of metadata: a first load job and a delta load job. The first load job performs a full replication, with any user-defined filters applied, from the source datastore to the target datastore. The delta load job can be used to capture only changed data after the first load.

The Workbench creates two scripts within the delta load job: the pre-load script and the post-load script. These scripts keep track of the time each job is executed so that the next execution can continue from where the last one ended. You can customize the pre-load and post-load scripts in the *Scripts* property tab for the replication job.

To help keep track of the delta job executions, the Workbench maintains an internal table in the target datastore that is created after the successful execution of the first load job. Because of this, the full replication job must be successfully executed at least once before the delta load job can be executed.

ℹ Note

By default, all target tables within the Workbench delta load job are created with the *Auto correct load* option enabled. For more information about this option, see the “Target tables” section in the *Reference Guide*.

ℹ Note

The delta load mode is disabled in the replication job execution if the selected target type is **Files**.

Delta job execution

During execution, the delta job first determines the last time that it was executed, if ever. It uses the last execution time in a filter within the data flows to select only those records that are new. After the data flows have completed successfully, the delta job updates the internal metadata with the current time, so that any future delta execution starts from that time.

Related Information

Reference Guide: Objects, Descriptions of objects, Target tables [page 963]

5.3.2.2 Delta load options for tables

For each table, the Workbench supports several delta load options:

ℹ Note

The delta load option will be disabled when a target type of **Files** is selected. In this case, only the No delta load option will be enabled.

- No delta load
  The Workbench skips this table in the delta load.
- Reload the full table
  The Workbench performs a complete refresh of the table.
- Use timestamp column
  The Workbench uses a timestamp column or columns from the table to perform the delta load. Data Services uses the timestamp from each load to extract only new data from after the last execution.
  If a table has more than one datetime or timestamp column, you must choose the column to use for the delta load process. If a table does not contain a datetime or timestamp column, this option is disabled. If you use a column that contains only the date, you can also choose to use an additional column that specifies the time.
- Use CDC
  The Workbench uses the changed data capture (CDC) functionality provided by a supported SAP application to extract only the data that has changed.

If the datastore does not support CDC functionality, this option is disabled.

- Use custom filter
  The Workbench uses a user-defined filter to determine the start time used to perform the delta load.

> ➜ Tip
>
> In the Workbench, the delta load job works best with tables that use a timestamp column as the primary key.

To change the delta load method for a table, right-click the table in the target schema pane and choose the delta load method under *Change delta load support*. You can also select the table in the target schema pane and choose the delta load method in the *Delta load* tab of the replication job editor.

5.3.2.3 Delta load options for SAP applications

For applications that support changed data capture (CDC) functionality, the Workbench automatically extracts changed data for the delta load.

You can use the Workbench-generated variable `$END_TIME` to specify the time frame for which to extract changed data:

- If the value of `$END_TIME` is not specified, the Workbench uses the current system time for its value. Each delta load will extract new data from the last execution to the current date-time.
- If you want to continue from a previous execution, retrieve the timestamp of the execution you want to recover from your application and assign it to `$END_TIME` in the replication job pre-load script.

> i Note
>
> The Workbench treats applications that do not support CDC as tables for the delta load.

5.4 Using an SAP Information Steward Data Cleansing Solution

The Workbench Data Cleansing Solution feature allows a technical ETL developer with no data quality expertise to create a data flow containing a data cleansing solution and validate the results in a production environment.

First, a data steward uses the Data Cleansing Advisor feature of SAP Information Steward to create and publish a data cleansing solution. The data cleansing solution is stored in the SAP BusinessObjects Business Intelligence platform's Central Management Server (CMS). In the Workbench, the ETL developer then models the data cleansing solution as a Data Cleansing Solution transform within the context of a data flow, and maps the transform's input and output schema to production sources and targets, and validates the results in the production environment. Finally, the ETL developer generates and deploys the data flow as ATL to the Data Services repository for execution in the Data Services engine.

For more information about data modeling in the Workbench, see the *SAP Data Services Workbench Guide*. For more information about input and output fields and options, see the *SAP Data Services Reference Guide*. For more information about Data Cleansing Advisor, see the *SAP Information Steward User Guide*.

5.4.1 Viewing data cleansing solutions

You can create multiple data flows that use a data cleansing solution. Each data flow can contain one or more unique instances of the Data Cleansing Solution transform that references the data cleansing solution.

The Data Cleansing Solution transform references the latest published version of the data cleansing solution from Data Cleansing Advisor. If a data cleansing solution is updated and republished in Data Cleansing Advisor, Workbench automatically uses the updated solution, and you must redeploy the data flow to the Data Services repository.

1. Select ▶ *Window* ▶ *Data Cleansing Solutions* ◀. The Data Cleansing Solutions tab opens in the workspace and displays the data cleansing solutions that have been published in the Data Quality Advisor.
   The Published column contains the date and time that the solution was published. Workbench displays only the latest active version of a data cleansing solution published in the Data Cleansing Advisor. Icons display the status of the data cleansing solution.

Icon	Description
▦	Indicates a new data cleansing solution that has not been used.
▦	Indicates a current data cleansing solution that has been used in one or more data flows.
▦	Indicates an updated data cleansing solution that has been republished since it was used.

2. To refresh the list of data cleansing solution, click the refresh icon (↻).
3. Click a column header to sort the data cleansing solutions by the content in that column.
4. To view more details about and identify the appropriate data cleansing solution, double-click a solution.
   The Data Cleansing Solution Details window opens and displays information about the solution such as the publication date and time, cleansing and matching settings, and the connection information and input schema that were used to create the data cleansing solution in Information Steward.
5. To open all of the data flows that contain a data cleansing solution, right-click the solution and select *Open Containing Data Flows* from the menu.

5.4.1.1 Deleting a data cleansing solution

You can delete a data cleansing solution in Workbench only if its corresponding project has first been deleted in the Data Cleansing Advisor in Information Steward.

To delete a data cleansing solution, right-click the solution and select *Delete* from the menu.

5.4.2 Configuring a data flow

Before you start modeling, a data cleansing solution must have already been created in SAP Information Steward and published to the CMS, and you should create a project, datastore, and data flow in the Workbench.

1. Open or create a data flow. The palette of elements on the right side of the window contains readers, loaders, and Platform and Data Integrator transforms.
2. Add a reader and loader to the data flow, as well as any other transforms required to manipulate the data.
3. Select ▸ *Window* ▸ *Data Cleansing Solutions* ◂.
   The Data Cleansing Solutions tab opens in the workspace and displays the data cleansing solutions that have been published in the Data Cleansing Advisor.
4. Select a data cleansing solution and drag it into the graphical area of the Data Flow editor. A Data Cleansing Solution transform is placed in the data flow that represents the data quality transforms, such as Data Cleanse, Global Address Cleanse, and Match. The underlying transforms included in the Data Cleansing Solution transform depend on the data cleansing solution recommendations that were chosen in Information Steward.
   The object properties panel opens below the data flow graphical area with three tabs.

Property tab	Description
Input	Displays a list of all input fields provided by the data cleansing solution that must be mapped in order for the data cleansing solution to be executed in Data Services. The specific input fields vary depending on the configuration of the data cleansing solution. The Description column displays a text description of the input field based on the content type that was identified by Data Cleansing Advisor.
Properties	Displays the Data Services options that you can set for the Data Cleansing Solution transform. Data quality configuration options (the business rules of cleansing and matching) are not editable within the Workbench. If an option setting must be modified, this is done in the Data Cleansing Advisor feature in SAP Information Steward, and the data cleansing solution must be republished.
Output	Displays a list of all available output fields from the data cleansing solution. The available output fields may vary depending on the configuration of the data cleansing solution.

5. To perform basic cleansing on your data to prepare it for more accurate cleansing and matching results, add a Basic Cleanse transform to the data flow and select the appropriate options. Typically, this basic cleansing is performed in Information Steward rather than Data Services Workbench.
6. Connect the Data Cleansing Solution transform to the reader and loader or other transforms in the data flow.
7. To automatically map fields from the reader to the Data Cleansing Solution transform, select the Input tab and click the *Detect Mappings* button.
   The Detect Mapping window opens and displays the field names candidates for mapping.
8. To confirm all of the suggested field mappings, click the *Confirm All* button; otherwise, select individual field names and click *Confirm* or *Reject* as appropriate. Click *OK*.
9. To manually map fields, select a field in the Input Schema and drag it to the appropriate row in the Input tab.
10. In the Output tab, select the field names that you want to output from the Available Output Fields and drag them into the Selected Fields list. If you want to output passthrough fields containing the original data before it was processed, select them in the Input Schema and drag them into the Selected Fields list. Use the Up and Down buttons to reorder the fields as you want to output them.

For information about transform options and input and output fields, see the *SAP Data Services Reference Guide*.

Related Information

Using the Basic Cleanse transform [page 2083]

5.4.2.1 Using the Basic Cleanse transform

In Information Steward, the Data Cleansing Advisor lets the data steward set up textual changes (transformations) to apply to specified columns in the data. Transformations set up in the wizard can help prepare the data for more accurate cleansing and matching results. When the data cleansing solution is published and then used in a data flow in Workbench, the transformations are represented in the Data Cleansing Solution transform, and the options are not editable.

Although basic cleansing is typically set up by the data steward in Information Steward, in Workbench, you can also add a Basic Cleanse transform, a specialized type of Query transform, to your data flow to perform the same cleansing on your data.

When the data flow is converted to ATL and deployed to the Data Services repository, the Basic Cleanse transform is converted to a Query transform.

Option	Description
Remove control characters	Removes control characters from the selected column.
Remove double quotes	Removes double quotation marks (") that appear anywhere in the selected column.
Remove leading spaces	Removes spaces before the text in the selected column.
Remove single quotes	Removes single quotation marks (') that appear anywhere in the selected column.
Remove trailing spaces	Remove spaces after the text in the selected column.
Replace null values with <default>	Enter the text that you want to take the place of empty fields in the selected column.
Search and replace	In the *Replace* box, enter the text that you want to delete from the selected column. In the *with* box, enter the text that you want to take the place of the deleted text. Click *Add*. Repeat these steps for each string of text that you want to replace. Use the up and down arrow buttons if you need to reorder search-and-replace values.

5.4.3 Modifying the reference data location

The location of the reference data is always set to the substitution parameter, $$RefFilesAddressCleanse, and cannot be modified within Workbench. The substitution parameter is set to the default location of the reference data during the Data Services installation, and Workbench automatically finds the reference data if you haven't modified the default.

To use a different reference data location to run a Data Cleansing Solution transform within Workbench, use the Data Services Designer to log into the repository that you want to modify, and edit the substitution parameter, $$RefFilesAddressCleanse, to point to the desired reference file directory. All Data Cleansing Solution transforms in Workbench will use this new location when executing a data flow for the repository.

5.4.4 Modifying the reports substitution variables

Enabling and disabling of data quality reports is set by the substitution parameters, $$ReportsAddressCleanse and $$ReportsMatch, and cannot be modified within Workbench. This allows reporting to be controlled at runtime within Data Services rather than being set within Workbench.

To enable or disable data quality reports for Workbench, use the Data Services Designer to log into the repository that you want to modify, and edit the substitution parameters, $$ReportsAddressCleanse and $$ReportsMatch.

5.4.5 Validating a data flow

To validate a data flow, click the *Validate* button. The status bar displays the validation status and the number of errors and warnings, if any.

Workbench performs the following validation on the Data Cleansing Solution transform:

- The Data Cleansing Solution can be retrieved from the SAP BusinessObjects Business Intelligence platform's Central Management Server (CMS).
- All input fields specified by the Data Cleansing Solution are mapped.
- All output fields in the Data Cleansing Solution transform output schema have a unique field name.
- At least one output field is specified in the Data Cleansing Solution transform.

If a validation error is found, the error is displayed in the Problems tab. Double-click an error to open the relevant Property tab in the Data Flow editor.

5.4.6 Deploying a data flow

Deploying a data flow publishes it to the Data Services repository.

The Data Cleansing Solution transform references the latest published version of the data cleansing solution from Data Cleansing Advisor. If a data cleansing solution is updated and republished in Data Cleansing Advisor, Workbench automatically uses the updated solution, and you must redeploy the data flow to the Data Services repository.

1. With the data flow open in the Data Flow editor, click the *Deploy* button. If you haven't saved the data flow, a prompt opens asking if you want to save the changes. Click *Yes* to save the changes.
   The Choose Repository window opens.
2. Select the Data Services repository that you want to deploy to, and click *OK*. When deployment is complete, an information window opens stating that the data flow was successfully deployed on the repository.
3. Click *Close*.

Deploying a data flow generates ATL that represents the elements in the data flow and can be executed by Data Services. After you deploy the job to the Data Services repository, you can open, modify, and execute it as a new, separate job instance in the Data Services Designer. However, any changes that you make in Data Services Designer are not saved to the Workbench instance of the job.

5.4.7 Executing a data flow

The execution process validates a data flow, generates ATL, deploys it to a Data Services repository, and executes the data flow.

1. With the data flow open in the Data Flow editor, click the *Execute* button.
   If you haven't saved the data flow, a prompt opens asking if you want to save the changes.
2. Click *Yes* to save the changes.
   The Execute job wizard opens.
3. In Step 1 of the Execute Job wizard, select the Data Services repository that you want to use and click *Next*.
   The wizard proceeds to the next step.
4. In Step 2 of the Execute Job wizard, select the appropriate execution parameters and click *Finish*.
   The job executes, and the monitor editor opens and displays the job status, execution progress, and any applicable logs. Trace, monitor, and error logs are available. When the job is completed, the job status is displayed at the top.

5.4.8 Viewing data

To view the data before and after it is processed, you can view data in both the reader and loader transforms.

1. In the graphical area of the Data Flow editor, select the reader or loader.
2. In the Properties tab for the selected loader, click the *View Data* button. A data grid opens and displays the first 200 records.
3. To view more data rows, edit the *Max Rows* field and click the refresh button.

5.4.9 Modifying a Workbench job in Data Services Designer

Workbench allows limited changes to transform options and fields in the data flow. In Data Services Designer, you can modify a data flow that was created in Workbench more extensively; however, any changes that you make in the Designer cannot be exported to or viewed in the Workbench. In the Designer, you see the individual transforms (Data Cleanse, Global Address Cleanse, and Match) that were represented by the Data Cleansing Solution transform in Workbench. You can change options, input fields, and output fields for Data Cleanse and Global Address Cleanse. You cannot modify Match settings. You can edit connection between all of the transforms on the canvas as necessary.

1. In the *Local Object Library* of the Data Services Designer, click the *Jobs* tab, and expand *Batch Jobs* to display the Workbench data cleansing solution job.
2. Double-click the job. The job opens on the canvas and contains a work flow.

3. Double-click the work flow. The work flow opens and contains a data flow.
4. Double-click the data flow. The data flow opens. Rather than displaying the single Data Cleansing Solution transform, the data flow contains the data quality transforms that the Data Cleansing Solution transform represents, such as Data Cleanse, Global Address Cleanse, and Match. The data quality transforms used depends on the configuration of the data cleansing solution created in Information Steward.
5. Modify and execute the job as necessary.

Any changes that you make in Data Services Designer are not saved to the Workbench instance of the job.

5.5 Additional Information

5.5.1 Supported data migration sources and targets

Supported sources

The Workbench supports the following database and application types as migration sources:

- Data Services-supported databases
  - Attunity Connector
  - HP Neoview
  - DB2
  - SAP HANA
  - Informix
  - Microsoft SQL Server
  - MySQL
  - Netezza
  - ODBC
  - Oracle
  - SQL Anywhere
  - Sybase ASE
  - Sybase IQ
  - Teradata
- SAP Applications
- SAP BW Source

i Note

For more information about the available options for each supported datastore source type, see the *Reference Guide*.

Supported targets

The Workbench supports the following database types as migration targets:

- SAP HANA
- DB2
- Netezza
- Oracle
- Sybase IQ
- Teradata
- Sybase ASE
- SQL Anywhere
- Microsoft SQL Server

> Note
>
> For more information about the available options for each supported datastore target type, see the *Reference Guide*.

Related Information

Reference Guide: Objects, Descriptions of objects, Datastore [page 860]

5.5.2 Using the expression editor

The expression editor allows you to use variables, functions, and basic logic structures in many areas of a replication job. For example, you can use the expression editor to customize the mapping of a target column or the replication job's pre-load or post-load scripts.

5.5.2.1 Supported operators

This section describes the operators that can be used in Workbench expressions and filters.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division

Operator	Description
=	Comparison, equal to
<	Comparison, less than
<=	Comparison, less than or equal to
>	Comparison, greater than
>=	Comparison, greater than or equal to
!=	Comparison, not equal to
||	Concatenate
%	Modulus Returns the remainder when one number is divided by another.
AND	Logical AND
OR	Logical OR
NOT	Logical NOT
IS NULL	Comparison, is a NULL value
IS NOT NULL	Comparison, is not a NULL value

5.5.2.2 Including functions in expressions

In Data Services, functions take input values and produce a return value. Input values can be parameters passed into a data flow, values from a column of data, or variables defined inside a script. This section discusses functions and how to include them in a Workbench replication job.

5.5.2.2.1 Built-in functions

This section describes each built-in function available in Data Services.

The following tables list the names and descriptions of functions, grouped by each function's category in the Workbench expression editor.

For a complete listing of all built-in functions and their syntax, parameters, and return values, see “Functions and Procedures, Descriptions of built-in functions” in the *Reference Guide*.

Aggregate functions

Function	Description
avg	Calculates the average of a given set of values.
count	Counts the number of values in a table column.
count_distinct	Count the number of distinct non-null values in a table column.
max	Returns the maximum value from a list.
min	Returns the minimum value from a list.
sum	Calculates the sum of a given set of values.

Conversion functions

Function	Description
cast	Returns a value in the cast data type.
extract_from_xml	Extracts XML directly from single column in a database table, and converts it into its internal nested relational data model (NRDM).
interval_to_char	Returns a string representation of the interval.
julian_to_date	Converts a Julian value to a date.
load_to_xml	Generates XML text from NRDM and loads it into a single database column (Assumes the database supports XML text in its columns).
long_to_varchar	Converts a data type from long to varchar.
num_to_interval	Converts a numeric value to an interval.
to_char	Converts a date or numeric type to a string.
to_date	Converts a string to a date.
to_decimal	Converts a varchar to a decimal.
varchar_to_long	Converts a data type from varchar to long.

Cryptographic functions

Function	Description
decrypt_aes	Decrypts the input string using the user-specified passphrase and key length using the AES algorithm.

Function	Description
encrypt_aes	Encrypts the input string using the user-specified passphrase and key length using the AES algorithm.

Database functions

Function	Description
key_generation	Generates keys for the specified table, after determining the appropriate starting value.
sql	Runs a SQL operation in the specified database.
total_rows	Returns the number of rows in a particular table in a datastore.

Date functions

Function	Description
add_months	Adds a given number of months to a date.
concat_date_time	Returns a datetime from separate date and time inputs.
date_diff	Returns the difference between two dates or times.
date_part	Extracts a component of a given date.
day_in_month	Determines the day in the month on which the given date falls.
day_in_week	Determines the day in the week on which the given date falls.
day_in_year	Determines the day in the year on which the given date falls.
fiscal_day	Converts a given date into an integer value representing a day in a fiscal year.
isweekend	Indicates that a date corresponds to Saturday or Sunday.
julian	Converts a date to its integer Julian value, the number of days between the start of the Julian calendar and the date.
last_date	Returns the last date of the month for a given date.
month	Determines the month in which the given date falls.
quarter	Determines the quarter in which the given date falls.
sysdate	Returns the current date as listed by the Job Server's operating system.
systime	Returns the current time as listed by the Job Server's operating system.

Function	Description
week_in_month	Determines the week in the month in which the given date falls.
week_in_year	Determines the week in the year in which the given date falls.
year	Determines the year in which the given date falls.

Environment functions

Function	Description
get_env	Returns a value for the specified environmental variable.
get_error_filename	Returns the full path and file name for the error log.
get_monitor_filename	Returns the full path and file name for the monitor log.
get_trace_filename	Returns the full path and file name for the trace log.
is_set_env	Verifies if the specified environment variable is set.
set_env	Sets an environmental variable temporarily to a specified value.

Lookup functions

Function	Description
lookup	Finds a value in one table or file based on values in a second table or file.
lookup_ext	Finds data from a database table, flat file, or memory datastore table.

Math functions

Function	Description
abs	Returns the absolute value of an input number.
ceil	Returns the smallest integer value greater than or equal to an input number.
floor	Returns the largest integer value less than or equal to an input number.
In	Returns the natural logarithm of the given numeric expression.

Function	Description
log	Returns the base-10 logarithm of the given numeric expression.
mod	Returns the remainder when one number is divided by another.
power	Returns the value of the give expression to the specified power.
rand	Returns a random number between 0 and 1.
rand_ext	Returns a random number between 0 and 1.
round	Rounds a given number to the specified precision.
sqrt	Returns the square root of the given expression.
trunc	Truncates a given number to the specified precision.

Miscellaneous functions

Function	Description
base64_decode	Returns the source data after decoding the base64-encoded input.
base64_encode	Returns the base64-encoded data in the engine locale character set.
current_configuration	Returns the name of the datastore configuration in use at runtime.
current_system_configuration	Returns the name of the system configuration in use at runtime. If no system configuration is defined, returns a NULL value.
dataflow_name	Returns the data flow name in which this call exists. If the call is not in a data flow, returns NULL.
datastore_field_value	Retrieves the value of a specified datastore field.
db_database_name	Returns the database name of the datastore configuration in use at runtime.
db_owner	Returns the real owner name for the datastore configuration that is in use at runtime.
db_type	Returns the database type of the datastore configuration in use at runtime.
db_version	Returns the database version of the datastore configuration in use at runtime.
decode	Returns an expression based on the first condition in the specified list that evaluates to TRUE.
file_exists	Checks to see if a given file or directory exists.

Function	Description
gen_row_num	Returns an integer value beginning with 1 then incremented sequentially by 1 for each additional call. This function can be used to generate a column of row IDs.
gen_row_num_by_group	Returns group row number of the record.
gen_uuid	Returns a unique varchar string identifier.
get_domain_description	Returns the description of a value when given the domain name and the value.
get_file_attribute	Returns date created, date modified, or size (in bytes) of a physical file.
greatest	Returns greatest of the list of one or more expressions.
host_name	Returns the name of the computer on which the job is executing.
ifthenelse	Allows conditional logic in mapping and selection operations.
is_group_changed	Returns 1 if the group is changed, 0 otherwise.
isempty	Indicates if a nested table contains data.
job_name	Returns the name of the job in which the call to this function exists.
job_run_id	Retrieves the job run ID for the current job execution.
least	Returns the least in a list of one ormore expressions.
nvl	Replaces NULL values.
previous_row_value	Returns the column value of previous row.
pushdown_sql	Allows you to create dynamic WHERE clauses.
raise_exception	Calling this function causes an exception to be generated.
raise_exception_ext	Same as raise_exception, but takes a second parameter for an exit code.
repository_name	Returns a database connection string and owner name. For example: beq-local.DBUser. This is the ID for the repository from which the job is run.
sleep	Suspends the execution of the data flow or work flow from where it is called.
system_user_name	Returns the user name used to log into the Job Server's operating system.
table_attribute	Retrieves the value of a specified table attribute.
truncate_table	Allows you to explicitly expunge data from a memory table.
wait_for_file	Returns the existing files that match the input file pattern.
workflow_name	Returns the work flow in which this call exists. Returns the name of the inner most work flow in cases where several work flows enclose this function call. If no work flow is found, returns job name.

SAP functions

Function	Description
sap_openhub_processchain_execute	Starts the process chain that extracts data from an SAP NetWeaver Business Warehouse(BW) and loads the extracted data into an Open Hub Destination table.
sap_openhub_set_read_status	Sends the read status for the Open Hub table to SAP NetWeaver BW.

String functions

Function	Description
ascii	Returns the decimal value of the first character for the given string using ASCII character set. If the character passed is not a valid ASCII character, -1 is returned.
chr	Get character representation of provided ASCII value.
double_metaphone	Encodes the input string usin the Double Metaphone algorithm and returns a string.
index	Returns the index of a given word in a string.
init_cap	Changes the characters in a string to title case.
length	Returns the number of characters in a given string.
literal	Returns an input constant expression without interpolation. Allows you to assign a pattern to a variable without interpolation.
lower	Changes the characters in a string to lowercase.
lpad	Pads a string with characters froma specified pattern.
lpad_ext	Pads a string with logical characters from a specified pattern.
ltrim	Removes specified characters from the start of a string.
ltrim_blanks	Removes blank characters from the start of a string.
ltrim_blanks_ext	Removes blank and control characters from the start of a string.
match_pattern	Matches whole input strings to simple patterns supported by Data Services. This function does not match substrings.
match_regex	Matches whole input strings to the pattern that you specify with regular expressions (regular expressions based on the POSIX standard) and flags. This function does not match substrings.
match_simple	
print	Prints the given string to the trace log.

Function	Description
replace_substr	Returns a string where every occurrence of a given search string in the input is substituted by the given replacement string.
replace_substr_ext	Takes an input string, replaces specified occurrences of a specified sub-string with a specified replacement and returns the result. You can also use this function to search for hexadecimal or reference characters.
rpad	Pads a string with characters from a given pattern.
rpad_ext	Pads a string with logical characters from a given pattern.
rtrim	Removes given characters from the end of a string.
rtrim_blanks	Removes blank characters from the end of a string.
rtrim_blanks_ext	Removes blank and control characters from the end of a string.
search_replace	Searches input parameters and replaces by matching criteria and values specified by search table.
soundex	Encodes the input string using the Soundex algorithm and returns a string. Use when you want to push down the function to the database level.
substr	Returns a specific portion of a string starting at a given point in the string.
upper	Changes the characters in a string to uppercase.
word	Returns one word out of a string.
word_ext	Returns the word identified by its position in a delimited string.

System functions

Function	Description
exec	Sends a command to the operating system for execution.
mail_to	Sends the specified e-mail message.
smtp_to	Sends the specified e-mail message using the SMTP protocol.

User functions

You can create your own functions by writing script functions in SAP Data Services scripting language using the Designer smart editor.

For more information about user-defined custom functions, see “Functions and Procedures, Custom functions” in the *Reference Guide*.

Validation functions

Function	Description
is_valid_date	Indicates if an expression can be converted into a valid date value.
is_valid_datetime	Indicates if an expression can be converted into a valid datetime value.
is_valid_decimal	Indicates if an expression can be converted into a valid decimal value.
is_valid_double	Indicates if an expression can be converted into a valid double value.
is_valid_int	Indicates if an expression can be converted into a valid integer value.
is_valid_real	Indicates if an expression can be converted into a valid real value.
is_valid_time	Indicates if an expression can be converted into a valid time value.

Related Information

Reference Guide: Functions and Procedures, Descriptions of built-in functions [page 1515]
Reference Guide: Functions and Procedures, Custom functions [page 1694]

5.5.3 Managing Workbench project information

5.5.3.1 Creating a Workbench project

If you want to create a new project, start the new project wizard in one of the following ways:

- Right-click inside the project explorer and choose *New* > *Project*.
- Choose *New* > *Project* from the *File* menu.
- Click the *New* toolbar icon. Choose *General* > *Project* and click *Next*.

1. Specify the name for the new project and click *Next*.

   Note

   The project name can contain only alphanumeric characters and the underscore character.

2. If you want to save the files for the project to a different location, uncheck *Use default location* and specify a location for the project files.
3. Select the Data Services repository.

4. Click *Finish* to create the project.

The new project is displayed in the project explorer.

You can now add Workbench objects such as datastores and replication jobs to your project.

5.5.3.2 Exporting resources and preferences

Use the Export wizard to export Workbench folders, resources, and user preferences.

1. Right-click within the project explorer pane and choose *Export*.
   The Export wizard appears.
2. Select the type of export that you want to perform and click *Next*.
   - Archive File
     Exports selected Workbench resources to a `.zip` or `.tar` archive on the local file system.
   - File System
     Exports selected Workbench resources directly to a location on the local file system.
   - Preferences
     Exports Workbench configuration preferences to a location on the local file system.
3. If you are exporting to an archive file or a location on the file system, select the resources to export and set the export options as desired.
   a) Select the folders and resources that you want to export.

   Folders appear in the left pane and resources appear in the right pane.

   > ➜ Tip
   >
   > Folders with a grayed-out selection box are not explicitly selected, but contain objects that will be exported.

   b) Click *Browse* to specify the location where the archive file or resource files will be created.
   c) If you are exporting to an archive file, select the type of archive file to create.
   d) Select the directory structure mode.

Option	Description
Create directory structure for files	Creates the directory structure for all selected resources, even if no folders are explicitly selected.
Create only selected directories	Creates the directory structure only for folders that are explicitly selected.

4. If you are exporting preferences, select the preferences to export and set the export options as desired.
   a) Select the preferences to export, or select *Export all* to export all Workbench preferences.
   b) Click *Browse* to specify the location where the preference file will be exported.
5. Click *Finish*.
   The selected resources or preferences are exported to the specified archive file or file system location.

5.5.3.3 Importing resources and preferences

Use the Import wizard to import Workbench folders, resources, and user preferences.

1. Right-click within the project explorer pane and choose *Import*.
   The Import wizard appears.
2. Select the type of export that you want to perform and click *Next*.
   - Archive File
     Imports selected Workbench resources from a `.zip` or `.tar` archive on the local file system.
   - File System
     Imports selected Workbench resource files from a location on the local file system.
   - Preferences
     Imports Workbench configuration preferences from a file on the local file system.
3. If you are importing from an archive file or a location on the file system, select the resources to import and set the import options as desired.
   a) Click *Browse* to specify the location of the archive file or resource files.
   b) Select the folders and resources that you want to import.

      Folders appear in the left pane and resources appear in the right pane.

      > ➜ Tip
      >
      > Folders with a grayed-out selection box are not explicitly selected, but contain objects that will be imported.

   c) Click *Browse* to specify the folder where the imported resources will be stored.
4. If you are importing preferences, select the preferences to import.
   a) Click *Browse* to specify the preference file to import.
   b) Select the preferences to import, or select *Import all* to import all Workbench preferences.
5. Click *Finish*.
   The selected resources or preferences are imported to the Workbench.

6 Performance Optimization Guide

6.1 Welcome to SAP Data Services

6.1.1 Welcome

SAP Data Services delivers a single enterprise-class solution for data integration, data quality, data profiling, and text data processing that allows you to integrate, transform, improve, and deliver trusted data to critical business processes. It provides one development UI, metadata repository, data connectivity layer, run-time environment, and management console—enabling IT organizations to lower total cost of ownership and accelerate time to value. With SAP Data Services, IT organizations can maximize operational efficiency with a single solution to improve data quality and gain access to heterogeneous sources and applications.

6.1.2 Documentation set for SAP Data Services

You should become familiar with all the pieces of documentation that relate to your SAP Data Services product. The latest Data Services documentation can be found on the *SAP Help Portal*.

Document	What this document provides
Adapter SDK Guide	Information about installing, configuring, and running the Data Services Adapter SDK
Administrator Guide	Information about administrative tasks such as monitoring, lifecycle management, security, and so on.
Customer Issues Fixed	Information about customer issues fixed in this release. i Note In some releases, this information is displayed the Release Notes.
Designer Guide	Information about how to use Data Services Designer.
Documentation Map	Information about available Data Services books, languages, and locations.
Installation Guide for Windows	Information about and procedures for installing Data Services in a Windows environment.
Installation Guide for UNIX	Information about and procedures for installing Data Services in a UNIX environment.
Integrator Guide	Information for third-party developers to access Data Services functionality using web services and APIs.
Master Guide	Information about the application, its components and scenarios for planning and designing your system landscape. Information about SAP Information Steward is also provided in this guide.

Document	What this document provides
Management Console Guide	Information about how to use Data Services Administrator and Data Services Metadata Reports.
Performance Optimization Guide	Information about how to improve the performance of Data Services.
Reference Guide	Detailed reference material for Data Services Designer.
Release Notes	Important information you need before installing and deploying this version of Data Services.
Technical Manuals	A compiled, searchable, “master” PDF of core Data Services books: ● *Administrator Guide* ● *Designer Guide* ● *Reference Guide* ● *Management Console Guide* ● *Performance Optimization Guide* ● *Integrator Guide* ● *Supplement for J.D. Edwards* ● *Supplement for Oracle Applications* ● *Supplement for PeopleSoft* ● *Supplement for Salesforce.com* ● *Supplement for Siebel* ● *Supplement for SAP* ● *Workbench Guide*
Text Data Processing Extraction Customization Guide	Information about building dictionaries and extraction rules to create your own extraction patterns to use with Text Data Processing transforms.
Text Data Processing Language Reference Guide	Information about the linguistic analysis and extraction processing features that the Text Data Processing component provides, as well as a reference section for each language supported.
Tutorial	A step-by-step introduction to using Data Services.
Upgrade Guide	Information to help you upgrade from previous releases of Data Services and release-specific product behavior changes from earlier versions of Data Services to the latest release.
What's New	Highlights of new key features in this SAP Data Services release. This document is not updated for support package or patch releases.
Workbench Guide	Provides users with information about how to use the Workbench to migrate data and database schema information between different database systems.

In addition, you may need to refer to several Supplemental Guides.

Document	What this document provides
Supplement for SAP	Information about interfaces between Data Services, SAP Applications, SAP Master Data Services, SAP NetWeaver BW, and SAP Master Data Services.

Document	What this document provides
Supplement for SuccessFactors	Information about interfaces between Data Services and SuccessFactors.
Supplement for Salesforce.com	Information about how to install, configure, and use the SAP Data Services Salesforce.com Adapter Interface.
Supplement for J.D. Edwards	Information about interfaces between Data Services and J.D. Edwards World and J.D. Edwards OneWorld.
Supplement for Oracle Applications	Information about the interface between Data Services and Oracle Applications.
Supplement for PeopleSoft	Information about interfaces between Data Servicesand PeopleSoft.
Supplement for Siebel	Information about the interface between Data Services and Siebel.

We also include these manuals for information about SAP BusinessObjects Information platform services.

Document	What this document provides
Information platform services Administrator Guide	Information for administrators who are responsible for configuring, managing, and maintaining an Information platform services installation.
Information platform services Installation Guide for UNIX	Installation procedures for SAP BusinessObjects Information platform services on a UNIX environment.
Information platform services Installation Guide for Windows	Installation procedures for SAP BusinessObjects Information platform services on a Windows environment.

6.1.3 Accessing documentation

You can access the complete documentation set for SAP Data Services in several places.

6.1.3.1 Accessing documentation on Windows

After you install SAP Data Services, you can access the documentation from the Start menu.

1. Choose *Start* > *Programs* > *SAP Data Services 4.2* > *Data Services Documentation* > *All Guides* .
2. Click the appropriate shortcut for the document that you want to view.

6.1.3.2 Accessing documentation on UNIX

After you install SAP Data Services, you can access the documentation by going to the directory where the printable PDF files were installed.

1. Go to **`<LINK_DIR>`**`/doc/book/en/`.

2. Using Adobe Reader, open the PDF file of the document that you want to view.

6.1.3.3 Accessing documentation from the Web

You can access the complete documentation set for SAP Data Services from the SAP Business Users Support site.

To do this, go to *http://help.sap.com/bods*.

You can view the PDFs online or save them to your computer.

6.1.4 SAP information resources

A global network of SAP technology experts provides customer support, education, and consulting to ensure maximum information management benefit to your business.

Useful addresses at a glance:

Address	Content
Customer Support, Consulting, and Education services *http://service.sap.com/*	Information about SAP support programs, as well as links to technical articles, downloads, and online forums. Consulting services can provide you with information about how SAP can help maximize your information management investment. Education services can provide information about training options and modules. From traditional classroom learning to targeted e-learning seminars, SAP can offer a training package to suit your learning needs and preferred learning style.
Product documentation *http://help.sap.com/bods/*	SAP product documentation.
Supported Platforms (Product Availability Matrix) *https://service.sap.com/PAM*	Get information about supported platforms for SAP Data Services. Use the search function to search for Data Services. Click the link for the version of Data Services you are searching for.
SAP Data Services Community Network *http://scn.sap.com/community/data-services*	Get online and timely information about SAP Data Services, including forums, tips and tricks, additional downloads, samples, and much more. All content is to and from the community, so feel free to join in and contact us if you have a submission.
Blueprints *http://scn.sap.com/docs/DOC-8820*	Blueprints for you to download and modify to fit your needs. Each blueprint contains the necessary SAP Data Services project, jobs, data flows, file formats, sample data, template tables, and custom functions to run the data flows in your environment with only a few modifications.

Address	Content
SAPTerm *https://portal.wdf.sap.corp/go/sapterm* ➦	SAP's terminology database, the central repository for defining and standardizing the use of specialist terms.

6.2 Environment Test Strategy

This section covers suggested methods of tuning source and target database applications, their operating systems, and the network used by your SAP Data Services environment. It also introduces key job execution options.

This section contains the following topics:

- *The source OS and database server* [page 2103]
- *The target OS and database server* [page 2104]
- *The network* [page 2104]
- *Job Server OS and job options* [page 2105]

To test and tune jobs, work with all four of these components in the order shown above.

In addition to the information in this section, you can use your UNIX or Windows operating system and database server documentation for specific techniques, commands, and utilities that can help you measure and tune the SAP Data Services environment.

6.2.1 The source OS and database server

Tune the source operating system and database to quickly read data from disks.

6.2.1.1 Operating system

Make the input and output (I/O) operations as fast as possible. The read-ahead protocol, offered by most operating systems, can greatly improve performance. This protocol allows you to set the size of each I/O operation. Usually its default value is 4 to 8 kilobytes which is too small. Set it to at least 64K on most platforms.

6.2.1.2 Database

Tune your database on the source side to perform SELECTs as quickly as possible.

In the database layer, you can improve the performance of SELECTs in several ways, such as the following:

- Create indexes on appropriate columns, based on your data flows.
- Increase the size of each I/O from the database server to match the OS read-ahead I/O size.
- Increase the size of the shared buffer to allow more data to be cached in the database server.
- Cache tables that are small enough to fit in the shared buffer. For example, if jobs access the same piece of data on a database server, then cache that data. Caching data on database servers will reduce the number of I/O operations and speed up access to database tables.

See your database server documentation for more information about techniques, commands, and utilities that can help you measure and tune the the source databases in your jobs.

6.2.2 The target OS and database server

Tune the target operating system and database to quickly write data to disks.

6.2.2.1 Operating system

Make the input and output operations as fast as possible. For example, the asynchronous I/O, offered by most operating systems, can greatly improve performance. Turn on the asynchronous I/O.

6.2.2.2 Database

Tune your database on the target side to perform INSERTs and UPDATES as quickly as possible.

In the database layer, there are several ways to improve the performance of these operations.

Here are some examples from Oracle:

- Turn off archive logging.
- Turn off redo logging for all tables.
- Tune rollback segments for better performance.
- Place redo log files and data files on a raw device, if possible.
- Increase the size of the shared buffer.

See your database server documentation for more information about techniques, commands, and utilities that can help you measure and tune the the target databases in your jobs.

6.2.3 The network

When reading and writing data involves going through your network, its ability to efficiently move large amounts of data with minimal overhead is very important. Do not underestimate the importance of network tuning (even if you have a very fast network with lots of bandwidth).

Set network buffers to reduce the number of round trips to the database servers across the network. For example, adjust the size of the network buffer in the database client so that each client request completely fills a small number of network packets.

6.2.4 Job Server OS and job options

Tune the Job Server operating system and set job execution options to improve performance and take advantage of self-tuning features of SAP Data Services.

6.2.4.1 Operating system

SAP Data Services jobs are multi-threaded applications. Typically a single data flow in a job initiates one `al_engine` process that in turn initiates at least 4 threads.

For maximum performance benefits:

- Consider a design that will run one `al_engine` process per CPU at a time.
- Tune the Job Server OS so that threads spread to all available CPUs.

For more information, see *Checking system utilization* [page 2108].

6.2.4.2 Jobs

You can tune job execution options after:

- Tuning the database and operating system on the source and the target computers
- Adjusting the size of the network buffer
- Your data flow design seems optimal

You can tune the following execution options to improve the performance of your jobs:

- *Monitor sample rate*
- *Collect statistics for optimization* and *Use collected statistics*

6.2.4.2.1 Setting Monitor sample rate

During job execution, the SAP Data Services writes information to the monitor log file and updates job events after the number of seconds specified in *Monitor sample rate* has elapsed. The default value is 5. Increase *Monitor sample rate* to reduce the number of calls to the operating system to write to the log file.

When setting *Monitor sample rate*, you must evaluate performance improvements gained by making fewer calls to the operating system against your ability to view more detailed statistics during job execution. With a higher *Monitor sample rate*, the software collects more data before calling the operating system to open the file, and performance improves. However, with a higher monitor rate, more time passes before you can view statistics during job execution.

> Note
>
> If you use a virus scanner on your files, exclude the SAP Data Services log from the virus scan. Otherwise, the virus scan analyzes the log repeatedly during the job execution, which causes a performance degradation.

6.2.4.2.2 Collecting statistics for self-tuning

SAP Data Services provides a self-tuning feature to determine the optimal cache type (in-memory or pageable) to use for a data flow.

6.2.4.2.3 Taking advantage of the self-tuning feature

1. When you first execute a job, select the option *Collect statistics for optimization* to collect statistics which include number of rows and width of each row. Ensure that you collect statistics with data volumes that represent your production environment. This option is not selected by default.
2. The next time you execute the job, this option is selected by default.
3. When changes occur in data volumes, re-run your job with *Collect statistics for optimization* to ensure that the software has the most current statistics to optimize cache types.

Related Information

Using Caches [page 2129]

6.3 Measuring Performance

6.3.1 Data Services processes and threads

Data Services uses processes and threads to execute jobs that extract data from sources, transform the data, and load data into a data warehouse. The number of concurrently executing processes and threads affects the performance of Data Services jobs.

6.3.1.1 Processes

The processes used to run jobs are:

- al_jobserver
  The al_jobserver initiates one process for each Job Server configured on a computer. This process does not use much CPU power because it is only responsible for launching each job and monitoring the job's execution.
- al_engine
  For batch jobs, an al_engine process runs when a job starts and for each of its data flows. Real-time jobs run as a single process.
  The number of processes a batch job initiates also depends upon the number of:
  - parallel work flows
  - parallel data flows
  - sub data flows

Related Information

Analyzing log files for task duration [page 2110]

6.3.1.2 Threads

A data flow typically initiates one `al_engine` process, which creates one thread per data flow object. A data flow object can be a source, transform, or target. For example, two sources, a query, and a target could initiate four threads.

If you are using parallel objects in data flows, the thread count will increase to approximately one thread for each source or target table partition. If you set the *Degree of parallelism* (DOP) option for your data flow to a value greater than one, the thread count per transform will increase. For example, a DOP of `5` allows five concurrent threads for a Query transform. To run objects within data flows in parallel, use the following features:

- Table partitioning
- File multithreading
- Degree of parallelism for data flows

Related Information

Using Parallel Execution [page 2137]

6.3.2 Measuring performance of jobs

You can use several techniques to measure performance of SAP Data Services jobs.

6.3.2.1 Checking system utilization

The number of processes and threads concurrently executing affects the utilization of system resources (see *Data Services processes and threads* [page 2106]).

Check the utilization of the following system resources:

- CPU
- Memory
- Disk
- Network

To monitor these system resources, use the following tools:

For UNIX:

- top or a third party utility (such as glance for HPUX)

For Windows:

- Performance tab on the Task Manager

Depending on the performance of your jobs and the utilization of system resources, you might want to adjust the number of processes and threads. The following sections describe different situations and suggests features to adjust the number of processes and threads for each situation.

6.3.2.1.1 CPU utilization

SAP Data Services is designed to maximize the use of CPUs and memory available to run the job.

The total number of concurrent threads a job can run depends upon job design and environment. Test your job while watching multi-threaded processes to see how much CPU and memory the job requires. Make needed adjustments to your job design and environment and test again to confirm improvements.

For example, if you run a job and see that the CPU utilization is very high, you might decrease the DOP value or run less parallel jobs or data flows. Otherwise, CPU thrashing might occur.

For another example, if you run a job and see that only half a CPU is being used, or if you run eight jobs on an eight-way computer and CPU usage is only 50%, you can be interpret this CPU utilization in several ways:

- One interpretation might be that the software is able to push most of the processing down to source and/or target databases.
- Another interpretation might be that there are bottlenecks in the database server or the network connection. Bottlenecks on database servers do not allow readers or loaders in jobs to use Job Server CPUs efficiently.
  To determine bottlenecks, examine:
  - Disk service time on database server computers
    Disk service time typically should be below 15 milliseconds. Consult your server documentation for methods of improving performance. For example, having a fast disk controller, moving database server log files to a raw device, and increasing log size could improve disk service time.
  - Number of threads per process allowed on each database server operating system. For example:

    - On HPUX, the number of kernel threads per process is configurable. The CPU to thread ratio defaults to one-to-one. It is recommended that you set the number of kernel threads per CPU to between 512 and 1024.
    - On Solaris and AIX, the number of threads per process is not configurable. The number of threads per process depends on system resources. If a process terminates with a message like "Cannot create threads," you should consider tuning the job.
      For example, use the *Run as a separate process* option to split a data flow or use the Data_Transfer transform to create two sub data flows to execute sequentially. Since each sub data flow is executed by a different al_engine process, the number of threads needed for each will be 50% less than in your previous job design.
      If you are using the *Degree of parallelism* option in your data flow, reduce the number for this option in the data flow Properties window.
  - Network connection speed
    Determine the rate that your data is being transferred across your network.
    - If the network is a bottle neck, you might change your job execution distribution level from sub data flow to data flow or job to execute the entire data flow on the local Job Server.
    - If the capacity of your network is much larger, you might retrieve multiple rows from source databases using fewer requests.
- Yet another interpretation might be that the system is under-utilized. In this case, you might increase the value for the *Degree of parallelism* option and increase the number of parallel jobs and data flows.

Related Information

Using Parallel Execution [page 2137]
Using grid computing to distribute data flow execution [page 2160]
Using array fetch size [page 2205]

6.3.2.1.2 Memory

For memory utilization, you might have one of the following different cases:

- Low amount of physical memory.
  In this case, you might take one of the following actions:
  - Add more memory to the Job Server.
  - Redesign your data flow to run memory-consuming operations in separate sub data flows that each use a smaller amount of memory, and distribute the sub data flows over different Job Servers to access memory on multiple machines.
  - Redesign your data flow to push down memory-consuming operations to the database.

  For example, if your data flow reads data from a table, joins it to a file, and then groups it to calculate an average, the group by operation might be occurring in memory. If you stage the data after the join and before the group by into a database on a different computer, then when a sub data flow reads the staged data and continues with the group processing, it can utilize memory from the database server on a different computer. This situation optimizes your system as a whole.

- Large amount of memory but it is under-utilized.
  In this case, you might cache more data. Caching data can improve the performance of data transformations because it reduces the number of times the system must access the database.
  There are two types of caches available: in-memory and pageable.
- Paging occurs.
  Pageable cache is the default cache type for data flows. On Windows, UNIX, and Linux, the virtual memory available to the al_engine process is 3.5 gigabytes (500 megabytes of virtual memory is reserved for other engine operations, totaling 4GB). You can change this default limit by increasing the value of the `MAX_64BIT_PROCESS_VM_IN_MB` parameter in the `DSConfig.txt` file.
  If more memory is needed than these virtual memory limits, the software starts paging to continue executing the data flow.
  If your job or data flow requires more memory than these limits, you can take advantage of one of the following features to avoid paging:
  - Split the data flow into sub data flows that can each use the amount of memory set by the virtual memory limits.
    Each data flow or each memory-intensive operation within a data flow can run as a separate process that uses separate memory from each other to improve performance and throughput.
  - Push-down memory-intensive operations to the database server so that less memory is used on the Job Server computer.

Related Information

Splitting a data flow into sub data flows [page 2153]
Push-down operations [page 2116]
Data_Transfer transform [page 2156]
Using grid computing to distribute data flow execution [page 2160]
Caching data [page 2129]

6.3.2.2 Analyzing log files for task duration

The trace log shows the progress of an execution through each component (object) of a job. The following depiction of a trace log shows a separate process ID (Pid) and time stamp for the job, data flow, and each sub data flow.

Pid	Tid	Type	Time Stamp	Message
...		...	...	...
5696	5964	JOB	2/11/2012 11:56:37 PM	Processing job <Group_Orders_Job>.
...		...	...	...
4252	4044	DATAFLOW	2/11/2012 11:56:38 PM	Process to execute data flow

Pid	Tid	Type	Time Stamp	Message
				<Group_Orders_DF> is started.
1604	984	DATAFLOW	2/11/2012 11:56:42 PM	Process to execute sub data flow <Group_Orders_DF_1> is started.
...		...	...	...
5648	5068	DATAFLOW	2/11/2012 11:56:48 PM	Process to execute sub data flow <Group_Orders_DF_2> is started.
...		...	...	...

Trace logs also include messages about sub data flows, caches, and statistics.

Related Information

Splitting a data flow into sub data flows [page 2153]
Caching data [page 2129]
Reference Guide: Objects, Log [page 935]

6.3.2.3 Reading the Monitor Log for execution statistics

The Monitor log file indicates how many rows SAP Data Services produces or loads for a job. By viewing this log during job execution, you can observe the progress of row-counts to determine the location of bottlenecks. You can use the Monitor log to answer questions such as the following:

- What transform is running at any moment?
- How many rows have been processed so far?
  The frequency that the Monitor log refreshes the statistics is based on Monitor sample rate.
- How long does it take to build the cache for a lookup or comparison table? How long does it take to process the cache?
  If take long time to build the cache, use persistent cache.
- How long does it take to sort?
  If take long time to sort, you can redesign your data flow to push down the sort operation to the database.
- How much time elapses before a blocking operation sends out the first row?
  If your data flow contains resource-intensive operations after the blocking operation, you can add Data_Transfer transforms to push-down the resource-intensive operations.

You can view the Monitor log from the following tools:

- The Designer, as the job executes, when you click the Monitor icon.
- The Administrator of the Management Console, when you click the Monitor link for a job from the Batch Job Status page.

The Monitor log in the Designer shows the path for each object in the job, the number of rows processed, and the elapsed time for each object. The Absolute time column displays the total time from the start of the job to when the software completes the execution of the data flow object.

Related Information

Setting Monitor sample rate [page 2105]
Using persistent cache [page 2134]
Push-down operations [page 2116]
Data_Transfer transform for push-down operations [page 2122]
Reference Guide: Objects, Log [page 935]

6.3.2.4 Reading the Performance Monitor for execution statistics

The Performance Monitor displays execution information for each work flow, data flow, and sub data flow within a job. You can display the execution times in a table format. You can use the Performance Monitor to answer questions such as the following:

- Which data flows might be bottlenecks?
- How much time did a a data flow or sub data flow take to execute?
- How many rows did the data flow or sub data flow process?
- How much memory did a specific data flow use?

> Note
>
> Memory statistics (Cache Size column) display in the Performance Monitor only if you select the *Collect statistics for monitoring* option when you execute the job.

6.3.2.4.1 To view the Performance Monitor

1. Access the Management Console with one of the following methods:
   - In the Designer top menu bar, click *Tools* and select *Management Console*.
   - Click ▶ *Start* ▸ *Programs* ▸ *SAP Data Services* **`<x.x>`** ▸ *Data Services Management Console* ◀.
2. On the launch page, click *Administrator*.
3. Select ▶ *Batch* ▸ **`<repository>`** ◀
4. On the Batch Job Status page, find a job execution instance.
5. Under *Job Information* for an instance, click *Performance Monitor*.

Related Information

To monitor and tune in-memory and pageable caches [page 2135]

6.3.2.5 Reading Operational Dashboards for execution statistics

Operational dashboard reports contain job and data flow execution information for one or more repositories over a given time period (for example the last day or week). You can use operational statistics reports to answer some of the following questions:

- Are jobs executing within the allotted time frames?
- How many jobs succeeded or failed over a given execution period?
- How is the execution time for a job evolving over time?
- How many rows did the data flow process?

6.3.2.5.1 Comparing execution times for the same job over time

1. Open the Management Console via one of the following methods:
    - In the Designer top menu bar, choose *Tools* > *Management Console*.
    - Choose *Start* > *Programs* > *SAP Data Services <x>.<x>* > *Data Services Management Console*.
2. On the launch page, click *Operational Dashboard*.
3. Look at the graphs in Job Execution Statistic History or Job Execution Duration History to see if performance is increasing or decreasing.
4. On the Job Execution Duration History page, if there is a specific day that looks high or low compared to the other execution times, click that point on the graph to view the Job Execution Duration graph for all of the jobs that ran that day.
5. Click *View all history* to compare different executions of a specific job or data flow.
6. On the Job Execution History tab, you can select a specific job and number of days.
7. On the Data Flow Execution History tab, you can select a specific job and number of days, as well as search for a specific data flow.

Related Information

Management Console Guide: Operational Dashboard Reports [page 1975]

6.4 Tuning Overview

This section presents an overview of the different Data Services tuning options, with cross-references to subsequent chapters for more details.

6.4.1 Strategies to execute jobs

6.4.1.1 Maximizing push-down operations to the database server

SAP Data Services generates SQL SELECT statements to retrieve the data from source databases. The software automatically distributes the processing workload by pushing down as much as possible to the source database server.

Pushing down operations provides the following advantages:

- Use the power of the database server to execute SELECT operations (such as joins, Group By, and common functions such as decode and string functions). Often the database is optimized for these operations.
- Minimize the amount of data sent over the network. Fewer rows can be retrieved when the SQL statements include filters or aggregations.

You can also do a full push down from the source to the target, which means the software sends SQL INSERT INTO... SELECT statements to the target database. The following features enable a full push down:

- Data_Transfer transform
- Database links and linked datastores

Related Information

Maximizing Push-Down Operations [page 2116]

6.4.1.2 Improving throughput

Use the following features to improve throughput:

- Using caches for faster access to data
  You can improve the performance of data transformations by caching as much data as possible. By caching data in memory, you limit the number of times the system must access the database.
- Bulk loading to the target
  The software supports database bulk loading engines including the Oracle bulk load API. You can have multiple bulk load processes running in parallel.
- Other tuning techniques

- Source-based performance options
  - Join ordering
  - Minimizing extracted data
  - Using array fetch size
- Target-based performance options
  - Loading method
  - Rows per commit
- Job design performance options
  - Loading only changed data
  - Minimizing data type conversion
  - Minimizing locale conversion
  - Precision in operations

6.4.1.3 Using advanced tuning options

If your jobs have CPU-intensive and memory-intensive operations, you can use the following advanced tuning features to improve performance:

- Parallel processes—Individual work flows and data flows can execute in parallel if you do not connect them in the Designer workspace.
- Parallel threads—The software supports partitioned source tables, partitioned target tables, and degree of parallelism. These options allow you to control the number of instances for a source, target, and transform that can run in parallel within a data flow. Each instance runs as a separate thread and can run on a separate CPU.
- Server groups and distribution levels—You can group Job Servers on different computers into a logical component called a server group. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the computer with the lightest load at runtime. This functionality also provides a hot backup method. If one Job Server in a server group is down, another Job Server in the group processes the job.
  You can distribute the execution of data flows or sub data flows within a batch job across multiple Job Servers within a Server Group to better balance resource-intensive operations.

Related Information

Using Parallel Execution [page 2137]

Management Console Guide: Server Groups [page 1885]

Using grid computing to distribute data flow execution [page 2160]

6.5 Maximizing Push-Down Operations

For SQL sources and targets, SAP Data Services creates database-specific SQL statements based on the data flow diagrams in a job. The software generates SQL SELECT statements to retrieve the data from source databases. To optimize performance, the software pushes down as many SELECT operations as possible to the source database and combines as many operations as possible into one request to the database. It can push down SELECT operations such as joins, Group By, and common functions such as decode and string functions.

Data flow design influences the number of operations that the software can push to the database. Before running a job, you can view the SQL that is generated and adjust your design to maximize the SQL that is pushed down to improve performance.

You can use database links and the Data_Transfer transform to push down more operations.

6.5.1 Push-down operations

By pushing down operations to the source database, Data Services reduces the number of rows and operations that the engine must retrieve and process, which improves performance. When determining which operations to push to the database, Data Services examines the database and its environment.

6.5.1.1 Full push-down operations

The Optimizer always first tries to do a full push-down operation. A full push-down operation is when all transform operations can be pushed down to the databases and the data streams directly from the source database to the target database. SAP Data Services sends SQL INSERT INTO... SELECT statements to the target database where SELECT retrieves data from the source.

The software does a full push-down operation to the source and target databases when the following conditions are met:

- All of the operations between the source table and target table can be pushed down.
- The source and target tables are from the same datastore, they are in datastores that have a database link defined between them, or if the datastore has linked remote servers.

To enable a full push-down from the source to the target, you can also use the following features:

- Data_Transfer transform
- Database links
- Use Linked Remote Servers option

For database targets that support the *Allow merge or upsert* option, when all other operations in the data flow can be pushed down to the source database, the auto-correct loading operation may also be pushed down for a full push-down operation to the target. The software sends an SQL MERGE INTO `<target>` statement that implements the *Ignore columns with value* and *Ignore columns with null* options.

6.5.1.2 Partial push-down operations

When a full push-down operation is not possible, SAP Data Services still pushes down the SELECT statement to the source database. Operations within the SELECT statement that the software can push to the database include:

- Aggregations — Aggregate functions, typically used with a *Group by* statement, always produce a data set smaller than or the same size as the original data set.
- Distinct rows — When you select *Distinct rows* from the *Select* tab in the query editor, the software will only output unique rows.
- Filtering — Filtering can produce a data set smaller than or equal to the original data set.
- Joins — Joins typically produce a data set smaller than or similar in size to the original tables. The software can push down joins when either of the following conditions exist:
  - The source tables are in the same datastore
  - The source tables are in datastores that have a database link defined between them
- Ordering — Ordering does not affect data-set size. The software can efficiently sort data sets that fit in memory. It is recommended that you push down the Order By for very large data sets.
- Projection — Projection is the subset of columns that you map on the *Mapping* tab in the query editor. Projection normally produces a smaller data set because it only returns columns needed by subsequent operations in a data flow.
- Functions — Most functions that have equivalents in the underlying database are appropriately translated. These functions include decode, aggregation, and string functions.

6.5.1.3 Operations that cannot be pushed down

SAP Data Services cannot push some transform operations to the database. For example:

- Expressions that include functions that do not have database correspondents
- Load operations that contain triggers
- Transforms other than Query
- Joins between sources that are on different database servers that do not have database links defined between them.

Similarly, the software cannot always combine operations into single requests. For example, when a stored procedure contains a COMMIT statement or does not return a value, the software cannot combine the stored procedure SQL with the SQL for other operations in a query.

The software can only push operations supported by the DBMS down to that DBMS. Therefore, for best performance, try not to intersperse SAP Data Services transforms among operations that can be pushed down to the database.

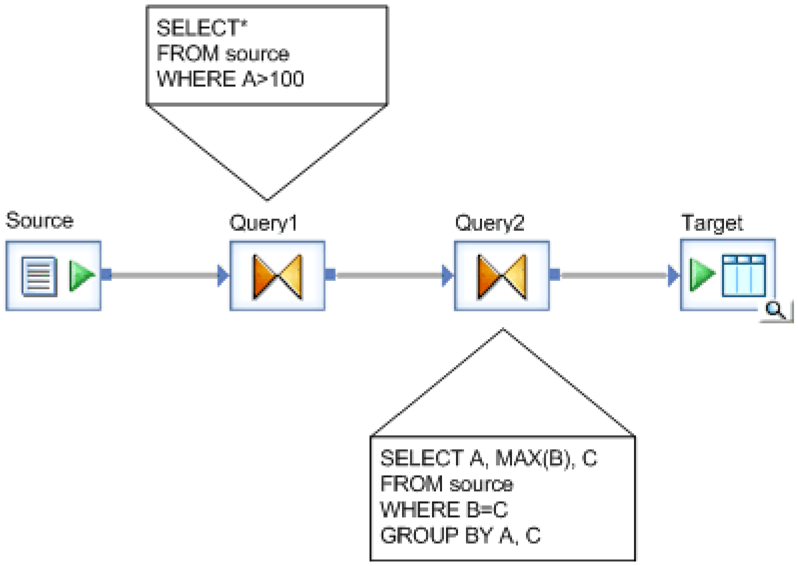
6.5.2 Push-down examples

The following are typical push-down scenarios.

6.5.2.1 Collapsing transforms to push down operations example

When determining how to push operations to the database, SAP Data Services first collapses all the transforms into the minimum set of transformations expressed in terms of the source table columns. Next, the software pushes all possible operations on tables of the same database down to that DBMS.

For example, the following data flow extracts rows from a single source table.



The first query selects only the rows in the source where column A contains a value greater than 100. The second query refines the extraction further, reducing the number of columns returned and further reducing the qualifying rows.

The software collapses the two queries into a single command for the DBMS to execute. The following command uses AND to combine the WHERE clauses from the two queries:

```
SELECT A, MAX(B), C
      FROM source
      WHERE A > 100 AND B = C
      GROUP BY A, C
```

The software can push down all the operations in this SELECT statement to the source DBMS.

6.5.2.2 Full push down from the source to the target example

If the source and target are in the same datastore, the software can do a full push-down operation where the INSERT into the target uses a SELECT from the source. In the sample data flow in scenario 1, a full push down passes the following statement to the database:

```
INSERT INTO target (A, B, C)
   SELECT A, MAX(B), C
```

```
       FROM source
       WHERE A > 100 AND B = C
       GROUP BY A, C
```

If the source and target are not in the same datastore, the software can also do a full push-down operation if you use one of the following features:

- Add a Data _Transfer transform before the target.
- Define a database link between the two datastores.

6.5.2.3 Full push down for auto correct load to the target example

For supported databases, if you enable the *Auto correct load* and *Allow merge or upsert* options, the Optimizer may be able to do a full push-down operation where the SQL statement is a MERGE into the target with a SELECT from the source.

In order for the *Allow merge or upsert* option to generate a MERGE statement, the primary key of the source table must be a subset of the primary key of the target table and the source row must be unique on the target key. In other words, there cannot be duplicate rows in the source data. If this condition is not met, the Optimizer pushes down the operation using a database-specific method to identify, update, and insert rows into the target table.

For example, suppose you have a data flow where the source and target tables are in the same datastore and the *Auto correct load* and *Allow merge or upsert* options are set to *Yes*.

The push-down operation passes the following statement to an Oracle database:

```
MERGE INTO "ODS"."TARGET" s
USING
SELECT "SOURCE"."A" A , "SOURCE"."B" B , "SOURCE"."C" C
                FROM "ODS"."SOURCE" "SOURCE"
                ) n
ON ((s.A = n.A))
WHEN MATCHED THEN
UPDATE SET s."B" = n.B,
                s."C" = n.C
WHEN NOT MATCHED THEN
INSERT /*+ APPEND */ (s."A", s."B", s."C" )
VALUES (n.A , n.B , n.C)
```

Similar statements are used for other supported databases.

6.5.2.4 Partial push down to the source example

If the data flow contains operations that cannot be passed to the DBMS, the software optimizes the transformation differently than the previous two scenarios. For example, if Query1 called `func(A) > 100`, where `func` is a SAP Data Services custom function, then the software generates two commands:

- The first query becomes the following command which the source DBMS executes:

  ```
  SELECT A, B, C
      FROM source
       WHERE B = C
  ```

- The second query becomes the following command which SAP Data Services executes because `func` cannot be pushed to the database:

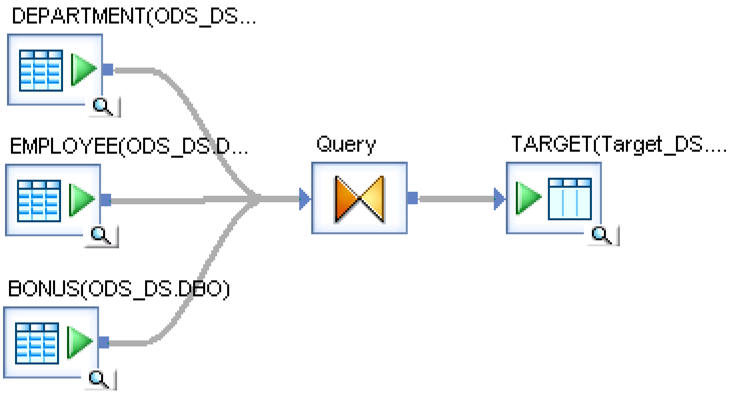
```
SELECT A, MAX(B), C
      FROM Query1
      WHERE func(A) > 100
    GROUP BY A, C
```

6.5.2.5 Push-down of SQL join example

If the tables to be joined in a query meet the requirements for a push-down operation, then the entire query is pushed down to the DBMS.

To confirm that the query will be pushed down, look at the Optimized SQL. If the query shows a single SELECT statement, then it will be pushed down.

For example, in the data flow shown below, the Department and Employee tables are joined with a inner join and then the result of that join is joined with left outer join to the Bonus table.



The resulting Optimized SQL contains a single select statement and the entire query is pushed down to the DBMS:

```
SELECT     DEPARTMENT.DEPTID, DEPARTMENT.DEPARTMENT, EMPLOYEE.LASTNAME,
BONUS.BONUS
FROM (DEPARTMENT INNER JOIN EMPLOYEE
(ON DEPARTMENT.DEPTID=EMPLOYEE.DEPTID))
LEFT OUTER JOIN BONUS
ON      (EMPLOYEE.EMPID = BONUS.EMPID)
```

Related Information

To view SQL [page 2121]

Maximizing Push-Down Operations [page 2116]

Reference Guide: Transforms, Query, Joins in the Query transform [page 1455]

6.5.3 To view SQL

Before running a job, you can view the SQL code that SAP Data Services generates for table sources in data flows. By examining the SQL code, you can verify that the software generates the commands you expect. If necessary, you can alter your design to improve the data flow.

1. Validate and save data flows.
2. Open a data flow in the workspace.
3. Select *Display Optimized SQL* from the *Validation* menu.

   Alternately, you can right-click a data flow in the object library and select Display Optimized SQL.

   The *Optimized SQL* window opens and shows a list of datastores and the optimized SQL code for the selected datastore. By default, the *Optimized SQL* window selects the first datastore.

   The software only shows the SELECT generated for table sources and INSERT INTO... SELECT for targets. It does not show the SQL generated for SQL sources that are not table sources, such as:

   - Lookup function
   - Key_generation function
   - Key_Generation transform
   - Table_Comparison transform
4. Select a name from the list of datastores on the left to view the SQL that this data flow applies against the corresponding database or application.

   The following example shows the optimized SQL for the second datastore which illustrates a full push-down operation (INSERT INTO... SELECT). This data flows uses a Data_Transfer transform to create a transfer table that the software loads directly into the target.

   ```
   INSERT INTO "DBO"."ORDER_AGG" ("SHIPCOUNTRY","SHIPREGION", "SALES_AGG")
   SELECT "TS_Query_Lookup"."SHIPCOUNTRY" ,
   "TS_Query_Lookup"."SHIPREGION" ,sum("TS_Query_Lookup"."SALES")
   FROM"DBO"."TRANS2""TS_Query_Lookup"
   GROUP BY "TS_Query_Lookup"."SHIPCOUNTRY" , "TS_Query_Lookup"."SHIPREGION"
   ```

   In the *Optimized SQL* window you can:

   - Use the *Find* button to perform a search on the SQL displayed.
   - Use the *Save As* button to save the text as a .sql file.

If you try to use the *Display Optimized SQL* command when there are no SQL sources in your data flow, the software alerts you. Examples of non-SQL sources include:

- Message sources
- File sources
- IDoc sources

If a data flow is not valid when you click the *Display Optimized SQL* option, the software alerts you.

> Note
>
> The *Optimized SQL* window displays the existing SQL statement in the repository. If you changed your data flow, save it so that the *Optimized SQL* window displays your current SQL statement.

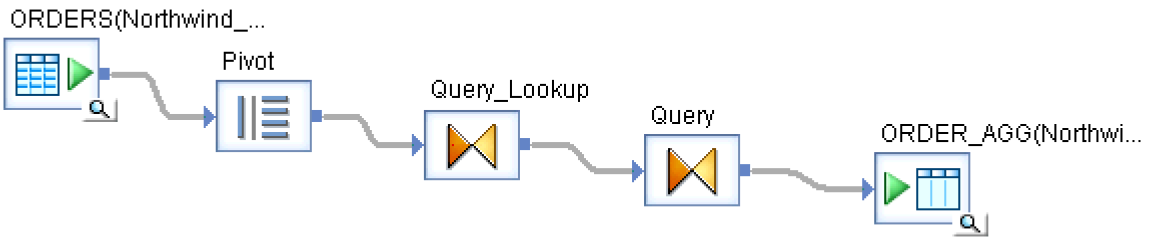
6.5.4 Data_Transfer transform for push-down operations

Use the Data_Transfer transform to move data from a source or from another transform into the target datastore and enable a full push-down operation (INSERT INTO... SELECT) to the target. You can use the Data_Transfer transform to push-down resource-intensive operations that occur anywhere within a data flow to the database. Resource-intensive operations include joins, GROUP BY, ORDER BY, and DISTINCT.

6.5.4.1 Push down an operation after a blocking operation example

You can place a Data_Transfer transform after a blocking operation to enable Data Services to push down a subsequent operation. A blocking operation is an operation that the software cannot push down to the database, and prevents ("blocks") operations after it from being pushed down.

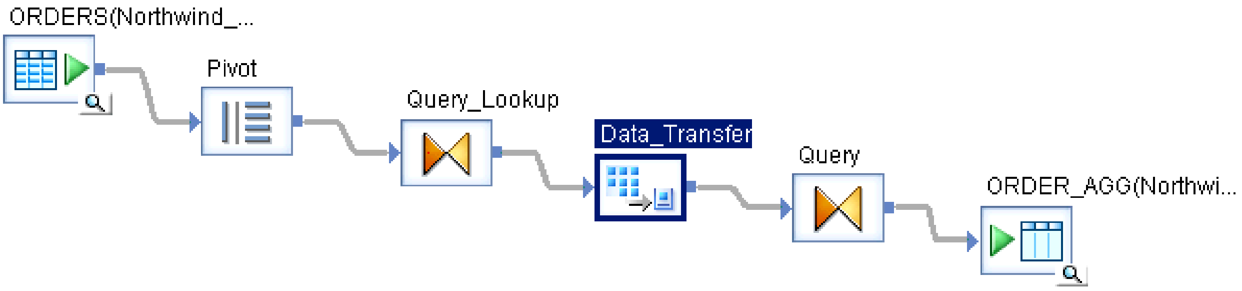
For example, you might have a data flow that groups sales order records by country and region, and sums the sales amounts to find which regions are generating the most revenue. The following diagram shows that the data flow contains a Pivot transform to obtain orders by Customer ID, a Query transform that contains a lookup_ext function to obtain sales subtotals, and another Query transform to group the results by country and region.



Because the Pivot transform and the lookup_ext function are before the query with the GROUP BY clause, the software cannot push down the GROUP BY operation. Here is how the *Optimized SQL* window would show the SELECT statement that the software pushes down to the source database:

```
SELECT "ORDERID", "CUSTOMERID", "EMPLOYEEID", "ORDERDATE", "REQUIREDDATE",
"SHIPPEDDATE",, "SHIPVIA"
"FREIGHT", "SHIPNAME", "SHIPADDRESS", "SHIPCITY", "SHIPREGION", "SHIPPOSTALCODE",
"SHIPCOUNTRY"
FROM "DBO"."ORDERS"
```

However, if you add a Data_Transfer transform before the second Query transform and specify a transfer table in the same datastore as the target table, the software can push down the GROUP BY operation.



The Data_Transfer Editor window shows that the transfer type is *Table* and the transfer table is in the same datastore as the target table (Northwind_DS.DBO.TRANS2).

Here's how the *Optimized SQL* window would show that the software pushed down the GROUP BY to the transfer table TRANS2.

```
INSTER INTO "DBO"."ORDER_AGG" ("SHIPCOUTNRY", "SHIPREGION", "SALES_AGG")
SELECT "TS_Query_Lookup"."SHIPCOUNTRY" , "TS_Query_Lookup"."SHIPREGION" ,
sum("TS_Query_Lookup"."SALES")
FROM "DBO"."TRANS2""TS_Query_Lookup"
GROUP BY "TS_Query_Lookup"."SHIPCOUNTRY" , "TS_Query_Lookup"."SHIPREGION"
```

Related Information

Reference Guide: Transforms, Data_Transfer [page 1071]

Operations that cannot be pushed down [page 2117]

6.5.4.2 Using Data_Transfer tables to speed up auto correct loads example

Auto correct loading ensures that the same row is not duplicated in a target table, which is useful for data recovery operations. However, an auto correct load prevents a full push-down operation from the source to the target when the source and target are in different datastores.

For large loads using database targets that support the *Allow merge or upsert* option for auto correct load, you can add a Data_Transfer transform before the target to enable a full push-down from the source to the target. In order for the *Allow merge or upsert* option to generate a MERGE statement:

- the primary key of the source table must be a subset of the primary key of the target table
- the source row must be unique on the target key

In other words, there cannot be duplicate rows in the source data. If this condition is not met, the Optimizer pushes down the operation using a database-specific method to identify, update, and insert rows into the target table.

If the MERGE statement can be used, SAP Data Services generates an SQL MERGE INTO **<target>** statement that implements the *Ignore columns with value* value (if a value is specified in the target transform editor) and the *Ignore columns with null* Yes/No setting.

For example, suppose you create a data flow that loads sales orders into an Oracle target table which is in a different datastore from the source.

For this data flow, the *Auto correct load* option is active and set to Yes, and the *Ignore columns with null* and *Allow merge or upsert* options are also active.

The SELECT statement that the software pushes down to the source database would look like the following (as it would appear in the *Optimized SQL* window).

```
SELECT "ODS_SALESORDER"."SALES_ORDER_NUMBER" , "ODS_SALESORDER"."ORDER_DATE" ,
"ODS_SALESORDER"."CUST_ID"
FROM "ODS"."ODS_SALESORDER""ODS_SALESORDER"
```

When you add a Data_Transfer transform before the target and specify a transfer table in the same datastore as the target, the software can push down the auto correct load operation.

The following MERGE statement is what the software pushes down to the Oracle target (as it appears in the *Optimized SQL* window).

```
MERGE INTO "TARGET"."AUTO_CORRECT_LOAD2_TARGET" s
USING
(SELECT "AUTOLOADTRANSFER"."SALES_ORDER_NUMBER" SALES_ORDER_NUMBER,
"AUTOLOADTRANSFER"."ORDER_DATE" ORDER_DATE, "AUTOLOADTRANSFER"."CUST_ID" CUST_ID
FROM "TARGET"."AUTOLOADTRANSFER" "AUTOLOADTRANSFER") n
ON ((s.SALES_ORDER_NUMBER=n.SALES_ORDRE_NUMBER00
WHEN MATCHED THEN
UPDATE SET s."ORDER_DATE"=nvl(n.ORDER_DATE,s."ORDER_DATE"),
s."CUST_ID"=nbl(n.CUST_ID,s."CUST_ID"
WHEN NOT MATCHED THEN
INSERT(s."SALES_ORDER_NUMBER",s."ORDER_DATE",s."CUST_ID")
VALUES(n.SALES_ORDRE_NUMBER,n.ORDRE_DATE,n.CUSTID)
```

Similar statements are used for other supported databases.

6.5.5 Database link and linked remote server support for push-down operations across datastores

Various database vendors support one-way communication paths from one database server to another. SAP Data Services refers to communication paths between databases as database links. The datastores in a database link relationship are called linked datastores, or, in the case of Sybase IQ, linked remote server.

The software uses linked datastores to enhance its performance by pushing down operations to a target database using a target datastore. Pushing down operations to a database not only reduces the amount of information that needs to be transferred between the databases and SAP Data Services but also allows the software to take advantage of the various DMBS capabilities, such as various join algorithms.

With support for database links, the software pushes processing down from different datastores, which can also refer to the same or different database type. Linked datastores allow a one-way path for data. For example, if you import a database link from target database B and link datastore B to datastore A, the software pushes the load operation down to database B, not to database A.

This section contains the following topics:

- Software support
- Example of push-down with linked datastores
- Example of push-down with linked remote servers
- Generated SQL statements
- Tuning performance at the data flow or Job Server level

Related Information

Designer Guide: Datastores, Linked datastores [page 231]

6.5.5.1 Software support

SAP Data Services supports push-down operations using linked datastores on all Windows and UNIX platforms. It supports DB2, Oracle, MS SQL server, and SAP Sybase SQL Anywhere databases.

Data Services supports push-down operations using linked remote server on all Windows and UNIX platforms. It supports Sybase IQ target and source datastore and Sybase ASE source datastore.

6.5.5.2 To take advantage of linked datastores

1. Create a database link on a database server that you intend to use as a target in a job.

   The following database software is required. See the Supported Platforms document for specific version numbers.

   - For DB2, use the DB2 Information Services (previously known as Relational Connect) software and make sure that the database user has privileges to create and drop a nickname.
     To end users and client applications, data sources appear as a single collective database in DB2. Users and applications interface with the database managed by the information server. Therefore, configure an information server and then add the external data sources. DB2 uses nicknames to identify remote tables and views.
     See the DB2 database manuals for more information about how to create links for DB2 and non-DB2 servers.
   - For Oracle, use the Transparent Gateway for DB2 and MS SQL Server.
     See the Oracle database manuals for more information about how to create database links for Oracle and non-Oracle servers.
   - For MS SQL Server, no special software is required.
     Microsoft SQL Server supports access to distributed data stored in multiple instances of SQL Server and heterogeneous data stored in various relational and non-relational data sources using an OLE database provider. SQL Server supports access to distributed or heterogeneous database sources in Transact-SQL statements by qualifying the data sources with the names of the linked server where the data sources exist.

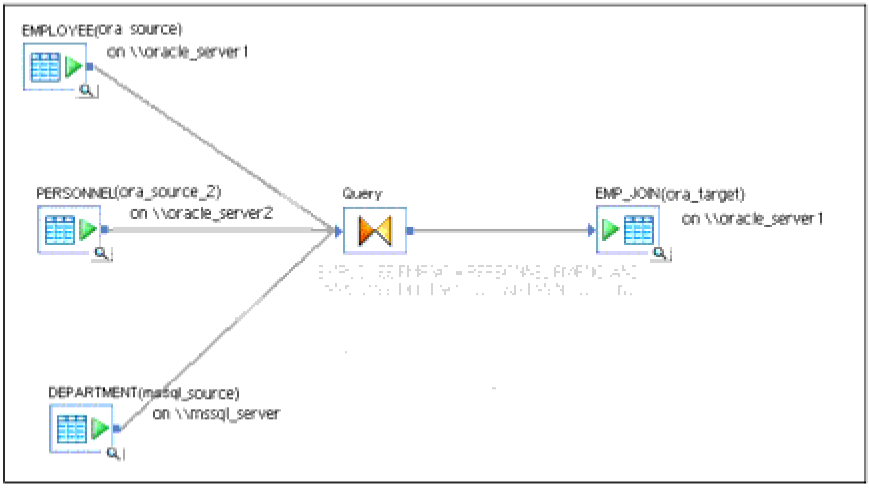
See the MS SQL Server database manuals for more information.

2. Create a database datastore connection to your target database.

6.5.5.2.1 Example of push-down with linked datastores

Linked datastores enable a full push-down operation (INSERT INTO... SELECT) to the target if all the sources are linked with the target. The sources and target can be in datastores that use the same database type or different database types.

The following diagram shows an example of a data flow that will take advantage of linked datastores:



The data flow joins three source tables from different database types:

- ora_source.HRUSER1.EMPLOYEE on \\oracle_server1
- ora_source_2.HRUSER2.PERSONNEL on \\oracle_server2
- mssql_source.DBO.DEPARTMENT on \\mssql_server3

The software loads the join result into the target table ora_target.HRUSER3.EMP_JOIN on \\oracle_server1.

In this data flow, the user (HRUSER3) created the following database links in the Oracle database oracle_server1.

Database Link Name	Local (to database link location) Connection Name	Remote (to database link location) Connection Name	Remote User
orasvr2	oracle_server1	oracle_server2	HRUSER2
tg4msql	oracle_server1	mssql_server	DBO

To enable a full push-down operation, database links must exist from the target database to all source databases and links must exist between the following datastores:

- ora_target and ora_source
- ora_target and ora_source2
- ora_target and mssql_source

The software executes this data flow query as one SQL statement in oracle_server1:

```
INSERT INTO HR_USER3.EMP_JOIN (FNAME, ENAME, DEPTNO, SAL, COMM)
SELECT psnl.FNAME, emp.ENAME, dept.DEPTNO, emp.SAL, emp.COMM
FROM HR_USER1.EMPLOYEE emp, HR_USER2.PERSONNEL@orasvr2 psnl,
oracle_server1.mssql_server.DBO.DEPARTMENT@tg4msql dept;
```

6.5.5.3 To take advantage of linked remote servers

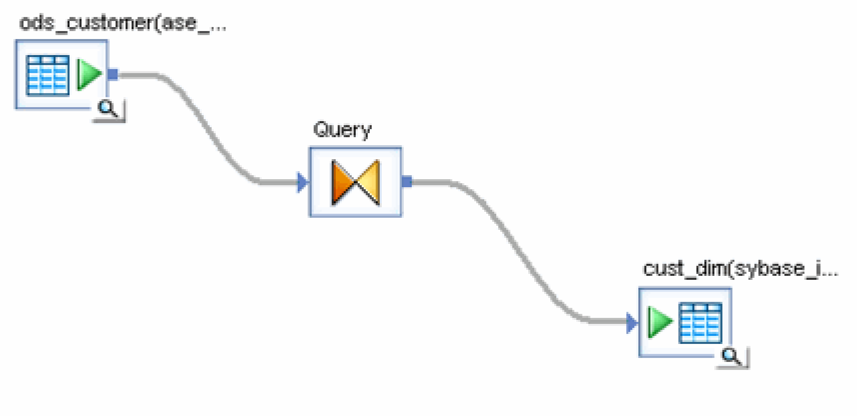
1. On the Sybase IQ server that you intend to use as a target for a job, create a remote server to the Sybase IQ or Sybase ASE server that you intend to use as a source for the job.
2. On the target server, create an external login to the same server that you configured the remote server for.
3. Refer to the Sybase documentation to set up the connectivity between the two servers that you want to communicate with.
4. Create a datastore connection to the target database and source database using the same server that you used to create the remote server.

Linked datastores enable a full push-down operation (INSERT INTO... SELECT) to a target if all of the sources are linked with the target. The sources and target can be in datastores that use the same database type or different database types.

6.5.5.3.1 Example of push-down with linked remote servers

Linked remote servers enable a full push-down operation (INSERT...LOCATION) to the target if all the target servers have remote links to the respective source server. Sources can be in Sybase IQ or Sybase ASE datastores and targets can be in Sybase IQ datastores.

The following diagram shows an example of a data flow that will take advantage of linked datastores. The source is a Sybase ASE datastore and the target is a Sybase IQ datastore with a remote server connection to Sybase ASE **<ase_remote>**.



The data flow selects data from a customer table and loads it to the cust_dim table using the INSERT...LOCATION SQL statement.

In this data flow, the user created a remote server link to the source table server Sybase_ase using the CREATE SERVER SQL statement in Sybase IQ. The user then configured a remote login for the server using the CREATE EXTERNLOGIN SQL statement. The remote login server name in the example is **`<ase_remote>`**.

The software executes the data flow query as one SQL statement in Sybase IQ:

```
INSERT INTO "DBA"."cust_dim" ( "Cust_ID" , "Cust_classf" , "Name1" , "Address" ,
"City" , "Region_ID" , "Zip" )
LOCATION 'ase_remote.cms57u05' PACKETSIZE 512 QUOTED IDENTIFIER ON ' SELECT
"ods_customer"."Cust_ID" , "ods_customer"."Cust_classf" , "ods_customer"."Name1" ,
"ods_customer"."Address" , "ods_customer"."City" , "ods_customer"."Region_ID" ,
"ods_customer"."Zip"
FROM "X9999"."ods_customer" "ods_customer"
```

6.5.5.4 Generated SQL statements

To see how SAP Data Services optimizes SQL statements, use *Display Optimized SQL* from the *Validation* menu when a data flow is open in the workspace.

- For DB2, it uses nicknames to refer to remote table references in the SQL display.
- For Oracle, it uses the following syntax to refer to remote table references: `<remote_table>@<dblink_name>`.
- For SQL Server, it uses the following syntax to refer to remote table references: `<linked_server>.<remote_database>.<remote_user>.<remote_table>`.

6.5.5.5 Tuning performance at the data flow or Job Server level

You might want to turn off linked-datastore push downs in cases where you do not notice performance improvements.

For example, the underlying database might not process operations from different data sources well. Data Services pushes down Oracle stored procedures and external functions. If these are in a job that uses database links, it will not impact expected performance gains. However, Data Services does not push down functions imported from other databases (such as DB2). In this case, although you may be using database links, Data Services cannot push the processing down.

Test your assumptions about individual job designs before committing to a large development effort using database links.

6.5.5.5.1 For a data flow

On the data flow properties dialog, this product enables the *Use database links* option by default to allow push-down operations using linked datastores. If you do not want to use linked datastores in a data flow to push down processing, deselect the check box.

This product can perform push downs using datastore links if the tables involved share the same database type and database connection name, or datasource name, even if the tables have different schema names. However, problems with enabling this feature could arise, for example, if the user of one datastore does not have access privileges to the tables of another datastore, causing a data access problem. In such a case, you can disable this feature.

6.5.5.5.2 For a Job Server

You can also disable linked datastores at the Job Server level. However, the *Use database links* option, at the data flow level, takes precedence.

Related Information

Designer Guide: Executing Jobs, Changing Job Server options [page 428]

6.6 Using Caches

6.6.1 Caching data

You can improve the performance of data transformations that occur in memory by caching as much data as possible. By caching data, you limit the number of times the system must access the database.

SAP Data Services provides the following types of caches that your data flow can use for all of the operations it contains:

- In-memory
  Use in-memory cache when your data flow processes a small amount of data that fits in memory.
- Pageable cache
  Use pageable cache when your data flow processes a very large amount of data that does not fit in memory. When memory-intensive operations (such as Group By and Order By) exceed available memory, the software uses pageable cache to complete the operation.

Pageable cache is the default cache type. To change the cache type, use the *Cache type* option on the data flow Properties window.

Note

If your data fits in memory, it is recommended that you use in-memory cache because pageable cache incurs an overhead cost.

6.6.1.1 Caching sources

By default, the *Cache* option is set to `Yes` in a source table or file editor to specify that data from the source is cached using memory on the Job Server computer. When sources are joined using the Query transform, the cache setting in the Query transform takes precedence over the setting in the source.

The default value for *Cache type* for data flows is `Pageable`.

It is recommended that you cache small tables in memory. Calculate the approximate size of a table with the following formula to determine if you should use a cache type of `Pageable` or `In-memory`.

```
table size = (in bytes)    # of rows *
                               # of columns *
                               20 bytes (average column size) *
                               1.3 (30% overhead)
```

Compute row count and table size on a regular basis, especially when:

- You are aware that a table has significantly changed in size.
- You experience decreased system performance.

If the table fits in memory, change the value of the *Cache type* option to `In-memory` in the Properties window of the data flow.

Related Information

Caching joins [page 2130]

6.6.1.2 Caching joins

The Cache setting indicates whether the software should read the required data from the source and load it into memory or pageable cache.

When sources are joined using the Query transform, the cache setting in the Query transform takes precedence over the setting in the source. In the Query editor, the cache setting is set to Automatic by default. The automatic setting carries forward the setting from the source table. The following table shows the relationship between cache settings in the source, Query editor, and whether the software will load the data in the source table into cache.

Cache Setting in Source	Cache Setting in Query Editor	Effective Cache Setting
Yes	Automatic	Yes
No	Automatic	No
Yes	Yes	Yes
No	Yes	Yes

Cache Setting in Source	Cache Setting in Query Editor	Effective Cache Setting
Yes	No	No
No	No	No

ℹ Note

If any one input schema has a cache setting other than Automatic specified in the Query editor, the Data Services Optimizer considers only Query editor cache settings and ignores all source editor cache settings.

ℹ Note

Best practice is to define the join rank and cache settings in the Query editor.

In the Query editor, cache a source only if it is being used as an inner source in a join.

When the cache setting is such that data will be cached if possible, a source is used as an inner source in a join under the following conditions:

- The source is specified as the inner source of a left outer join.
- In an inner join between two tables, the source has a lower join rank.

Caching does not affect the order in which tables are joined.

If optimization conditions are such that the software is pushing down operations to the underlying database, it ignores your cache setting.

If a table becomes too large to fit in the cache, ensure that the cache type is pageable.

Related Information

About join ordering [page 2201]

6.6.1.3 Changing cache type for a data flow

You can improve the performance of data transformations that occur in memory by caching as much data as possible. By caching data, you limit the number of times the system must access the database.

To change the cache type for a data flow:

1. In the object library, select the data flow name.
2. Right-click and choose *Properties*.
3. On the *General* tab of the *Properties* window, select the desired cache type in the drop-down list for the *Cache type* option.

6.6.1.4 Caching lookups

You can also improve performance by caching data when looking up individual values from tables and files.

6.6.1.4.1 Using a Lookup function in a query

SAP Data Services has three Lookup functions: `lookup`, `lookup_seq`, and `lookup_ext`. The `lookup` and `lookup_ext` functions have cache options. Caching lookup sources improves performance because the software avoids the expensive task of creating a database query or full file scan on each row.

You can set cache options when you specify a lookup function. There are three caching options:

Option	Description
NO_CACHE	Does not cache any values.
PRE_LOAD_CACHE	Preloads the result column and compare column into memory (it loads the values before executing the lookup).
DEMAND_LOAD_CACHE	Loads the result column and compare column into memory as the function executes. Use this option when looking up highly repetitive values that are a small subset of the data and when missing values are unlikely. Demand-load caching of lookup values is helpful when the lookup result is the same value multiple times. Each time the software cannot find the value in the cache, it must make a new request to the database for that value. Even if the value is invalid, the software has no way of knowing if it is missing or just has not been cached yet. When there are many values and some values might be missing, demand-load caching is significantly less efficient than caching the entire source.

6.6.1.4.2 Using a source table and setting it as the outer join

Although you can use lookup functions inside SAP Data Services queries, an alternative is to expose the translate (lookup) table as a source table in the data flow diagram, and use an outer join (if necessary) in the query to look up the required data. This technique has some advantages:

- You can graphically see the table the job will search on the diagram, making the data flow easier to maintain.
- The software can push the execution of the join down to the underlying RDBMS (even if you need an outer join).

This technique also has some disadvantages:

- You cannot specify default values in an outer join (default is always null), but you can specify a default value in lookup_ext.
- If an outer join returns multiple rows, you cannot specify what to return, (you can specify MIN or MAX in lookup_ext).

- The workspace can become cluttered if there are too many objects in the data flow.
- There is no option to use DEMAND_LOAD caching, which is useful when looking up only a few repetitive values in a very large table.

> ➜ Tip
>
> If you use the lookup table in multiple jobs, you can create a persistent cache that multiple data flows can access. For more information, see *Using persistent cache* [page 2134].

6.6.1.5 Caching table comparisons

You can improve the performance of a Table_Comparison transform by caching the comparison table. There are three modes of comparisons:

- Row-by-row select
- Cached comparison table
- Sorted input

Of the three, *Row-by-row select* will likely be the slowest and *Sorted input* the fastest.

> ➜ Tip
>
> If you want to sort the input to the table comparison transform, then choose the *Sorted input* option for comparison.

> ➜ Tip
>
> If the input is not sorted, then choose the *Cached comparison table* option.

6.6.1.6 Specifying a pageable cache directory

If the memory-consuming operations in your data flow exceed the available memory, SAP Data Services uses pageable cache to complete the operation. Memory-intensive operations include the following operations:

- Distinct
- Functions such as count_distinct and lookup_ext
- Group By
- Hierarchy_Flattening
- Order By

> i Note
>
> The default pageable cache directory is %LINKDIR\Log\PCache. If your data flows contain memory-consuming operations, change this value to a pageable cache directory that:

- Contains enough disk space for the amount of data you plan to profile.
- Is on a separate disk or file system from the SAP Data Services system.

Change the directory in the *Specify a directory with enough disk space for pageable cache* option in the Server Manager, under Runtime resources configured for this computer.

6.6.2 Using persistent cache

Persistent cache datastores provide the following benefits for data flows that process large volumes of data.

- You can store a large amount of data in persistent cache which SAP Data Services quickly pages into memory each time the job executes. For example, you can access a lookup table or comparison table locally (instead of reading from a remote database).
- You can create cache tables that multiple data flows can share (unlike a memory table which cannot be shared between different real-time jobs). For example, if a large lookup table used in a lookup_ext function rarely changes, you can create a cache once and subsequent jobs can use this cache instead of creating it each time.

Persistent cache tables can cache data from relational database tables and files.

> i Note
>
> You cannot cache data from hierarchical data files such as XML messages and SAP IDocs (both of which contain nested schemas). You cannot perform incremental inserts, deletes, or updates on a persistent cache table.

You create a persistent cache table by loading data into the persistent cache target table using one data flow. You can then subsequently read from the cache table in another data flow. When you load data into a persistent cache table, SAP Data Services always truncates and recreates the table.

6.6.2.1 Using persistent cache tables as sources

After you create a persistent cache table as a target in one data flow, you can use the persistent cache table as a source in any data flow. You can also use it as a lookup table or comparison table.

Related Information

Reference Guide: Objects, Persistent cache source [page 951]

6.6.3 Monitoring and tuning caches

This section describes the following topics:

Related Information

Using statistics for cache self-tuning [page 2135]
To monitor and tune in-memory and pageable caches [page 2135]

6.6.3.1 Using statistics for cache self-tuning

SAP Data Services uses cache statistics collected from previous job runs to automatically determine which cache type to use for a data flow. Cache statistics include the number of rows processed.

The default cache type is pageable. The software can switch to in-memory cache when it determines that your data flow processes a small amount of data that fits in memory.

6.6.3.1.1 Choosing the cache type automatically

1. Run your job with options *Collect statistics for optimization*.
2. Run your job again with option *Use collected statistics* (this option is selected by default).

6.6.3.2 To monitor and tune in-memory and pageable caches

You can also monitor and choose the cache type to use for the data flow.

1. Test run your job with options *Collect statistics for optimization* and *Collect statistics for monitoring*.

   > Note
   >
   > The option *Collect statistics for monitoring* is costly to run because it determines the cache size for each row processed.

2. Run your job again with option *Use collected statistics* (this option is selected by default).
3. Look in the Trace Log to determine which cache type was used.
   - The first time you run the job or if you have not previously collected statistics, the following messages indicate that cache statistics are not available and the sub data flows use the default cache type, pageable.

     ```
     Cache statistics for sub data flow <GroupBy_DF_1_1> are not available to be
     used for optimization and need to be collected before they can be used.
     ```

     ```
     Sub data flow <GroupBy_DF_1_1> using PAGEABLE Cache with <1280 MB> buffer
     pool.
     ```

   - You might see the following message that indicates that the software is switching to In-memory cache:

     ```
     Cache statistics determined that sub data flow <GroupBy_DOP2_DF_1_4> uses <1>
     caches with a total size of <1920> bytes. This is less than (or equal to) the
     ```

```
virtual memory <1342177280> bytes available for caches. Statistics is
switching the cache type to IN MEMORY.
```

```
Sub data flow <GroupBy_DOP2_DF_1_4> using IN MEMORY Cache.
```

Because pageable cache is the default cache type for a data flow, you might want to permanently change the *Cache type* to In-Memory in the data flow *Properties* window.

- You might see the following messages that indicate on sub data flow uses IN MEMORY cache and the other sub data flow uses PAGEABLE cache:

```
Sub data flow <Orders_Group_DF_1> using IN MEMORY Cache.
```

```
...
```

```
Sub data flow <Orders_Group_DF_2> using PAGEABLE Cache with <1536 MB> buffer
pool.
```

4. Look in the Administrator Performance Monitor to view data flow statistics and see the cache size.
   a) On the Administrator, select ▶ *Batch* > **<repository name>** ◀.
   b) On the Batch Job Status page, find a job execution instance.
   c) Under *Job Information* for an instance, click *Performance Monitor*. The Administrator opens the Table tab of the Performance Monitor page. This tab shows a tabular view of the start time, stop time, and execution time for each work flow, data flow, and sub data flow within the job.
   d) To display statistics for each object within a data flow or sub data flow, click one of the data flow names on the Table tab. The Transform tab displays the following statistics.

Statistic	Description
Name	Name that you gave the object (source, transform, or target) in the Designer.
Type	Type of object within the data flow. Possible values include Source, Mapping, Target.
Start time	Date and time this object instance started execution.
End time	Date and time this object instance stopped execution.
Execution time (sec)	Time (in seconds) the object took to complete execution.
Row Count	Number of rows that this object processed.
Cache Size (KB)	Size (in kilobytes) of the cache that was used to process this object. Note This statistics displays only if you selected *Collect statistics for monitoring* for the job execution.

5. If the value in Cache Size is approaching the physical memory limit on the job server, consider changing the *Cache type* of a data flow from *In-memory* to *Pageable*.

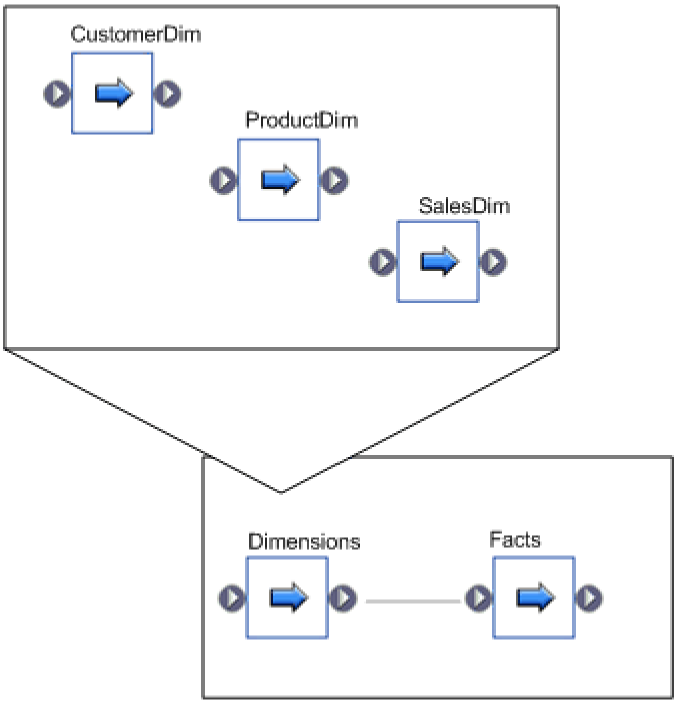
6.7 Using Parallel Execution

You can set SAP Data Services to perform data extraction, transformation, and loads in parallel by setting parallel options for sources, transforms, and targets. In addition, you can set individual data flows and work flows to run in parallel by simply not connecting them in the workspace. If the Job Server is running on a multi-processor computer, it takes full advantage of available CPUs.

6.7.1 Parallel data flows and work flows

You can explicitly execute different data flows and work flows in parallel by not connecting them in a work flow or job. SAP Data Services coordinates the parallel steps, then waits for all steps to complete before starting the next sequential step.

For example, use parallel processing to load dimension tables by calling work flows in parallel. Then specify that your job creates dimension tables before the fact table by moving it to the left of a second (parent) work flow and connecting the flows.



Parallel engine processes execute the parallel data flow processes. Note that if you have more than eight CPUs on your Job Server computer, you can increase *Maximum number of engine processes* to improve performance. To change the maximum number of parallel engine processes, use the Job Server options (*Tools* > *Options> Job Server* > *Environment*).

6.7.2 Parallel execution in data flows

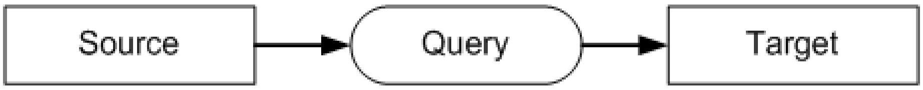
For batch jobs, SAP Data Services allows you to execute parallel threads in data flows.

6.7.2.1 Table partitioning

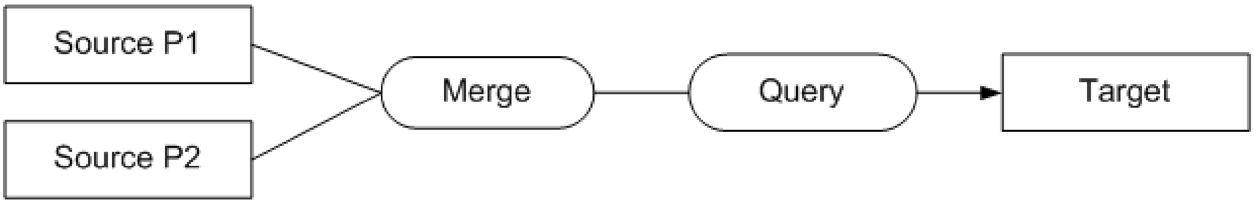
SAP Data Services processes data flows with partitioned tables based on the amount of partitioning defined.

6.7.2.1.1 Data flow with source partitions only

If you have a data flow with a source that has two partitions connected to a query and a target, it appears in the workspace as shown in the following diagram:



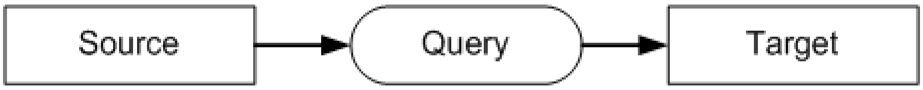
At runtime, the software translates this data flow to:



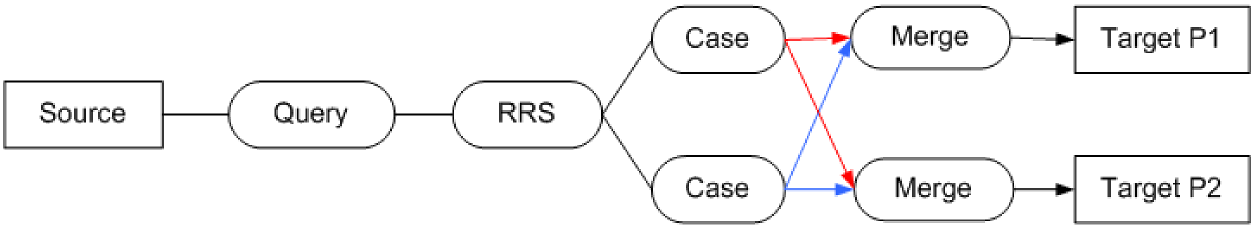
The software instantiates a source thread for each partition, and these threads run in parallel. The data from these threads later merges into a single stream by an internal merge transform before processing the query.

6.7.2.1.2 Data flow with target partitions only

If you have a data flow with a target that has two partitions connected to a query and a source, it appears in the workspace as shown in the following diagram:



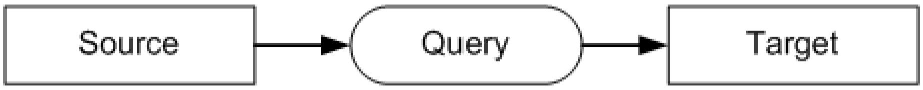
At runtime, the software translates this data flow to:



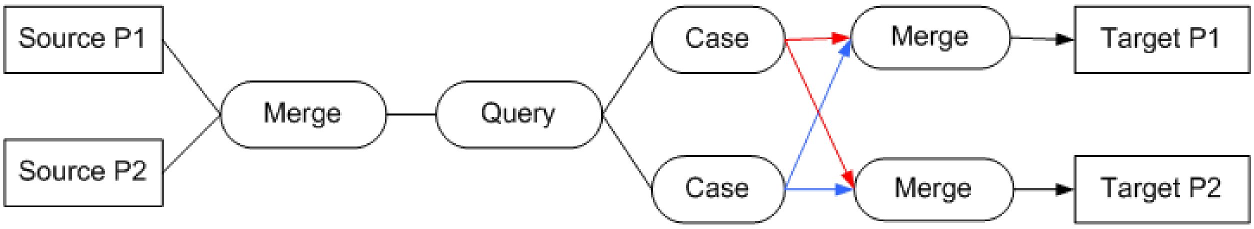
The software inserts an internal Round Robin Split (RRS) transform after the Query transform, which routes incoming rows in a round-robin fashion to internal Case transforms. The Case transforms evaluate the rows to determine the partition ranges. Finally, an internal Merge transform collects the incoming rows from different Case transforms and outputs a single stream of rows to the target threads. The Case transforms, Merge transform, and target threads execute in parallel.

6.7.2.1.3 data flow with source and target partitions

If you have a data flow with a source that has two partitions connected to a query and a target that has two partitions, it appears in the workspace as shown in the following diagram:



At runtime, the software translates this data flow to:



The source threads execute in parallel and the Case transforms, Merge transforms, and targets execute in parallel.

6.7.2.1.4 Viewing, creating, and enabling table partitions

Oracle databases support range, list, and hash partitioning. You can import this information as table metadata and use it to extract data in parallel. You can use range and list partitions to load data to Oracle targets. You can also specify logical range and list partitions using SAP Data Services metadata for Oracle tables.

In addition, it provides the ability to specify logical range and list partitions for DB2, Microsoft SQL Server, SAP R/3, SAP Sybase ASE, and SAP Sybase IQ tables by modifying imported table metadata.

SAP Data Services uses partition information by instantiating a thread at runtime for each partition. These threads execute in parallel. To maximize performance benefits, use a multi-processor environment.

6.7.2.1.4.1 Viewing partition information

1. Import a table into SAP Data Services.
2. In the *Datastores* tab of the object library, right-click the table name and select *Properties*.
3. Click the *Partitions* tab.

When you import partitioned tables from your database, you will find these partitions displayed on the Partitions tab of the table's Properties window. The partition name appears in the first column. The columns that are used for partitioning appear as column headings in the second row.

If you import a table that does not have partitions, you can create logical partitions using the Partitions tab of the table's Properties window.

6.7.2.1.4.2 Creating or editing table partition information

1. In the *Datastores* tab of the object library, right-click the table name and select *Properties*.
2. In the Properties window, click the *Partitions* tab.
3. Select a partition type.

Partition Type	Description
None	This table is not partitioned.
Range	Each partition contains a set of rows with column values less than those specified. For example, if the value of column one is `100,000`, then the data set for partition one will include rows with values less than `100,000` in column one.
List	Each partition contains a set of rows that contain the specified column values.

> **Note**
>
> If you imported an Oracle table with hash partitions, you cannot edit the hash settings in SAP Data Services. The Partitions tab displays the hash partition name and ID as read-only information. However, you can change the partition type to Range or List to create logical range or list partitions for an Oracle table imported with hash partitions.

4. Add, insert, or remove partitions and columns using the tool bar. (See the table at the end of this procedure.)
5. Select the name of a column from each column list box.
6. Enter column values for each partition.

   SAP Data Services validates the column values entered for each partition according to the following rules:

   - Values can be literal numbers and strings or datetime types.
   - Column values must match column data types.
   - Literal strings must include single quotes: `'Director'`.
   - For range partitions, the values for a partition must be greater than the values for the previous partition.
   - For the last partition, you can enter the value `MAXVALUE` to include all values.
7. Click *OK*.

   If the validation rules described in the previous step are not met, you will see an error message.

Icon	Description
	Add Partition
	Insert Partition
	Remove Partition
	Add Column
	Insert Column
	Remove Column

The number of partitions in a table equals the maximum number of parallel instances that the software can process for a source or target created from this table.

In addition to importing partitions or creating and editing partition metadata, enable the partition settings when you configure sources and targets.

6.7.2.1.4.3 Enabling partition settings in a source or target table

1. Drop a table into a data flow and select *Make Source* or *Make Target*.
2. Click the name of the table to open the source or target table editor.
3. Enable partitioning for the source or target:
   a) For a source table, click the *Enable Partitioning* check box.
   b) For a target table, click the *Options* tab, then click the *Enable Partitioning* check box.
4. Click *OK*.

   When the job executes, the softwarre generates parallel instances based on the partition information.

   > i Note
   >
   > If you are loading to partitioned tables, a job will execute the load in parallel according to the number of partitions in the table. If you set *Enable Partitioning* to Yes and *Include in transaction* to No, the *Include in transaction* setting overrides the *Enable Partitioning* option. For example, if your job is designed to load to a partitioned table but you set *Include in transaction* to Yes and enter a value for *Transaction order*, when the job executes, the software will include the table in a transaction load and does not parallel load to the partitioned table.

6.7.2.1.4.4 Tip

If the underlying database does not support range partitioning and if you are aware of a natural distribution of ranges, for example using an Employee Key column in an Employee table, then you can edit the imported table metadata and define table ranges. The software would then instantiate multiple reader threads, one for each defined range, and execute them in parallel to extract the data.

> **Note**
>
> Table metadata editing for partitioning is designed for source tables. If you use a partitioned table as a target, the physical table partitions in the database must match the metadata table partitions in SAP Data Services. If there is a mismatch, the software will not use the partition name to load partitions. Consequently, the whole table updates.

6.7.2.2 Degree of parallelism

T degree of parallelism (DOP) is a property of a data flow that defines how many times each transform defined in the data flow replicates for use on a parallel subset of data. If there are multiple transforms in a data flow, SAP Data Services chains them together until it reaches a merge point.

You can run transforms in parallel by entering a number in the *Degree of Parallelism* option on a data flow's Properties window. The number is used to replicate transforms in the data flow which run as separate threads when the Job Server processes the data flow.

6.7.2.2.1 Degree of parallelism and transforms

The Query transform always replicates when you set the DOP to a value greater than 1. SAP Data Services also replicates query operations such as Order By, Group By, join, and functions such as lookup_ext.

The Table Comparison replicates when you use the Row-by-row select and Cached comparison table comparison methods.

- Map_Operation
- History_Preserving
- Pivot

There are two basic scenarios:

- DOP and a data flow with a single transform
- DOP and a data flow with multiple transforms

DOP and a data flow with a single transform

The following figures show runtime instances of a data flow with a DOP of 1, and the same data flow with a DOP of 2.

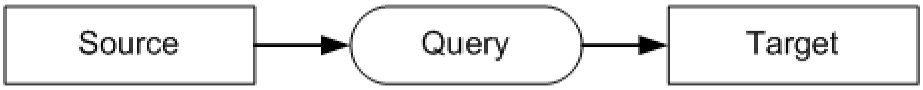


Figure 1: Runtime instance of a data flow where DOP =1

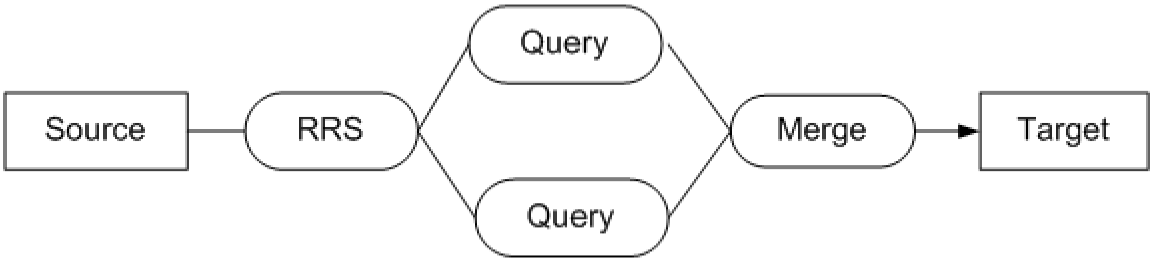


Figure 2: Runtime instance of a data flow where DOP = 2

With a DOP greater than 1, the software inserts an internal Round Robin Split (RRS) that transfers data to each of the replicated queries. The replicated queries execute in parallel, and the results merge into a single stream by an internal Merge transform.

DOP and a data flow with multiple transforms

The following figures show runtime instances of a data flow with a DOP of 1, and the same data flow with a DOP of 2. Notice multiple transforms in a data flow replicate and chain when the DOP is greater than 1.

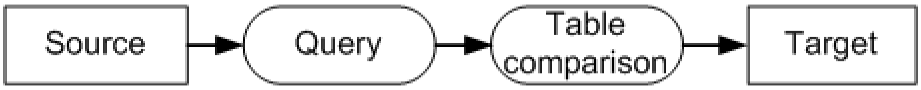


Figure 3: Runtime instance of a data flow where DOP =1

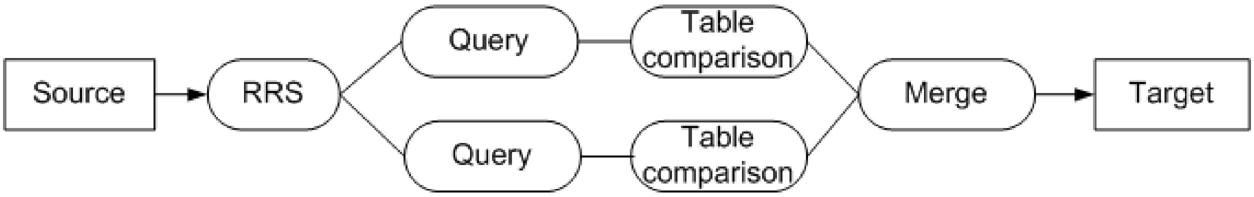


Figure 4: Runtime instance of a data flow where DOP = 2

When there are multiple transforms in a data flow and the DOP is greater than 1, the software carries the replicated stream as far as possible, then merges the data into a single stream.

6.7.2.2.2 To set the degree of parallelism for a data flow

The degree of parallelism (DOP) is a data flow property that acts on transforms added to the data flow.

1. In the object library, select the Data Flow tab.
2. Right-click the data flow icon and select *Properties*.

3. Enter a number in the *Degree of parallelism* option.

   The default value for degree of parallelism is 0. If you set an individual data flow's degree of parallelism to this default value, then you can control it using a Global_DOP value which affects all data flows run by a given Job Server. If you use any other value for a data flow's degree of parallelism, it overrides the Global_DOP value.

   You can use the local and global DOP options in different ways. For example:

   - If you want to globally set all data flow DOP values to 4, but one data flow is too complex and you do not want it to run in parallel, you can set the *Degree of parallelism* for this data flow locally. From the data flow's Properties window, set this data flow's *Degree of parallelism* to 1. All other data flows will replicate and run transforms in parallel after you set the Global_DOP value to 4. The default for the Global_DOP value is 2.
   - If you want to set the DOP on a case-by-case basis for each data flow, set the value for each data flow's *Degree of parallelism* to any value except zero.

   You set the Global_DOP value in the Job Server options.
4. Click *OK*.

Related Information

Designer Guide: Executing Jobs, Changing Job Server options [page 428]

6.7.2.2.3 Degree of parallelism and joins

If your Query transform joins sources, DOP determines the number of times the join replicates to process a parallel subset of data.

This section describes two scenarios:

- DOP and executing a join as a single process
- DOP and executing a join as multiple processes

DOP and executing a join as a single process

The following figures show runtime instances of a data flow that contains a join with a DOP of 1 and the same data flow with a DOP of 2. You use join ranks to define the outer source and inner source. In both data flows, the inner source is cached in memory.

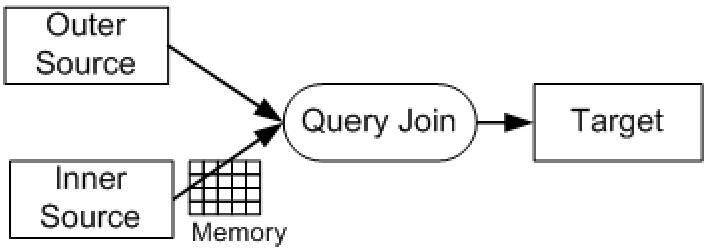


Figure 5: Runtime instance of a join where DOP =1

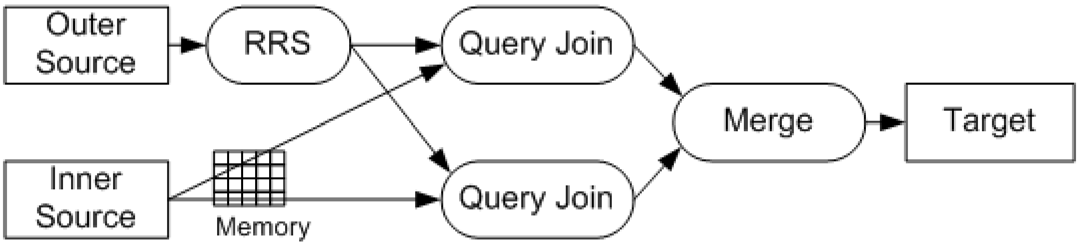


Figure 6: Runtime instance of a join where DOP = 2

With a DOP greater than one, the software inserts an internal Round Robin Split (RRS) that transfers data to each of the replicated joins. The inner source is cached once, and each half of the outer source joins with the cached data in the replicated joins. The replicated joins execute in parallel, and the results merge into a single stream by an internal Merge transform.

DOP and executing a join as multiple processes

When you select the *Run JOIN as a separate process* in the Query transform, you can split the execution of a join among multiple processes. the software creates a sub data flow for each separate process.

The following figure shows a runtime instance of a data flow that contains a join with a DOP of 2 and the *Run JOIN as a separate process* option selected.

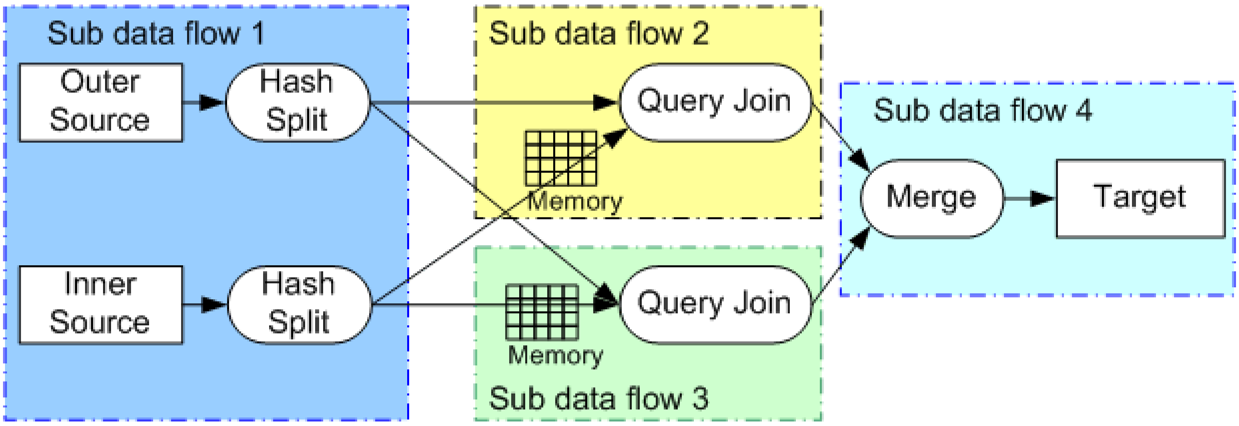


Figure 7: Runtime instance of a join that runs as multiple processes and DOP = 2

The data flow becomes four sub data flows (indicated by the blue dotted and dashed line in the figure):

- The first sub data flow uses an internal hash algorithm to split the data.
- The next two sub data flows are the replicated joins that run as separate processes.
- The last sub data flow merges the data and loads the target.

> ➔ Tip
>
> If DOP is greater than one, select either *job* or *data flow* for the *Distribution level* option when you execute the job. If you execute the job with the value *sub data flow* for *Distribution level*, the Hash Split sends data to the replicated queries that might be executing on different Job Servers. Because the data is sent on the network between different Job Servers, the entire data flow might be slower.

Related Information

About join ordering [page 2201]
Caching joins [page 2130]
Using grid computing to distribute data flow execution [page 2160]

6.7.2.2.4 Degree of parallelism and functions

You can set stored procedures and custom functions to replicate with the transforms in which they are used. To specify this option, select the *Enable parallel execution* check box on the function's Properties window. If this option is not selected and you add the function to a transform, the transform will not replicate and run in parallel even if its parent data flow has a *Degree of parallelism* value greater than `1`.

When enabling functions to run in parallel, verify that:

- Your database will allow a stored procedure to run in parallel.
- A custom function set to run in parallel will improve performance.

All built-in functions, except the following, replicate if the transform they are used in replicates due to the DOP value:

• `avg()`	• `min()`
• `count()`	• `previous_row_value()`
• `count_distinct()`	• `print()`
• `double_metaphone()`	• `raise_exception()`
• `exec()`	• `raise_exception_ext()`
• `get_domain_description()`	• `set_env()`
• `gen_row_num()`	• `sleep()`

- `gen_row_num_by_group()`	- `smtp_to()`
- `is_group_changed()`	- `soundex()`
- `key_generation()`	- `sql()`
- `mail_to()`	- `sum()`
- `max()`	- `total_rows()`

6.7.2.2.5 Enable stored procedures to run in parallel

Use the *Enable parallel execution* option to set functions to run in parallel when the transforms in which they are used execute in parallel.

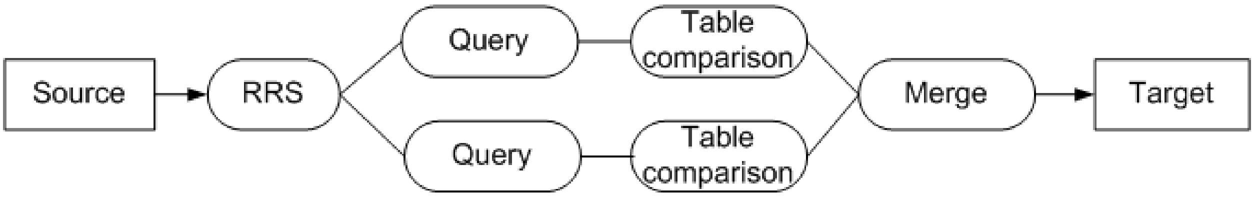
1. In the *Datastores* tab of the object library, expand a *Datastore* node.
2. Expand its *Function* node.
3. Right-click a function and select *Properties*.
4. In the Properties window, click the *Function* tab.
5. Click the *Enable Parallel Execution* check box.
6. Click *OK*.

6.7.2.2.5.1 To enable custom functions to run in parallel

1. In the *Custom Functions* tab of the object library, right-click a function name and select *Properties*.
2. In the Properties window, click the *Function* tab.
3. Click the *Enable Parallel Execution* check box.
4. Click *OK*.

6.7.2.2.6 Tips

DOP can degrade performance if you do not use it judiciously. The best value to choose depends on the complexity of the flow and number of CPUs available. For example, on a computer with four CPUs, setting a DOP greater than two for the following data flow will not improve performance but can potentially degrade it due to thread contention.



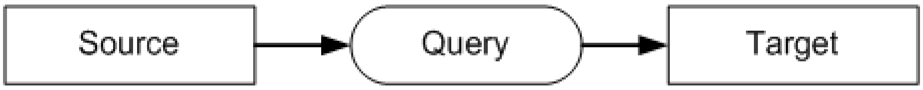
If your data flow contains an Order By or a Group By that is not pushed down to the database, put them at the end of a data flow. A sort node (Order By, Group By) is always a merge point, after which the engine proceeds as if the DOP value is 1.

Related Information

To view SQL [page 2121]

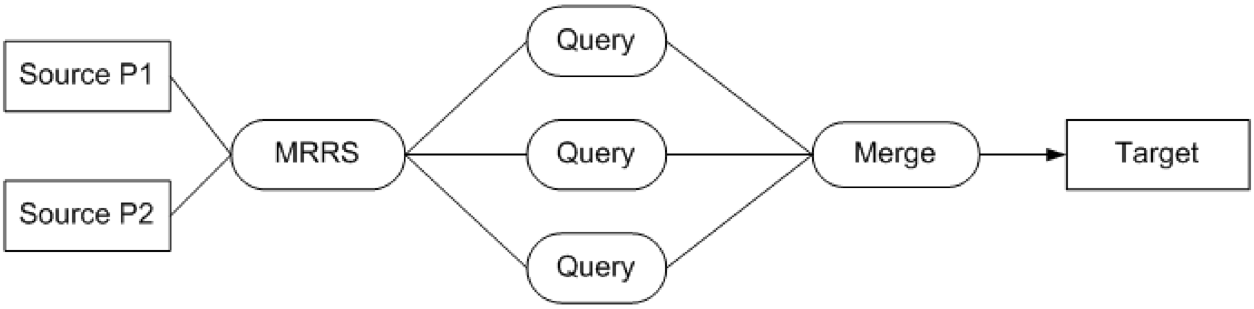
6.7.2.3 Combining table partitioning and a degree of parallelism

Different settings for source and target partitions and the degree of parallelism result in different behaviors in the SAP Data Services engine. The sections that follow show some examples. For all the following scenarios, the data flow appears as follows:



6.7.2.3.1 Two source partitions and a DOP of three

When a source has two partitions, it replicates twice. The input feeds into a merge-round-robin splitter (MRRS) that merges the input streams and splits them into a number equal to the value for DOP (in this case, three outputs to the query transform). The stream then merges and feeds into the target.

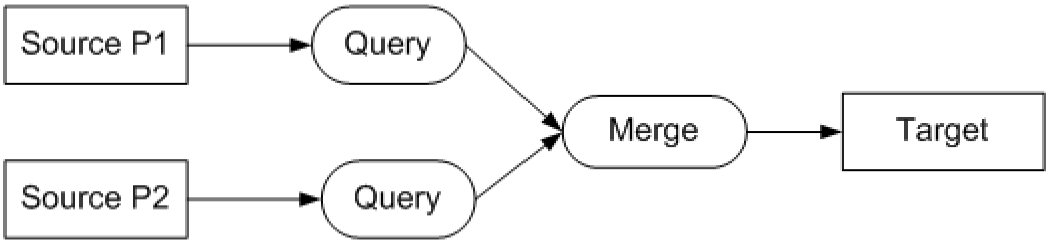


Tip

If the target is not partitioned, set the *Number of loaders* option equal to the DOP value. Depending on the number of CPUs available, set the DOP value equal to the number of source partitions as a general rule. This produces a data flow without the Merge Round Robin Split and each partition pipes the data directly into the consuming transform.

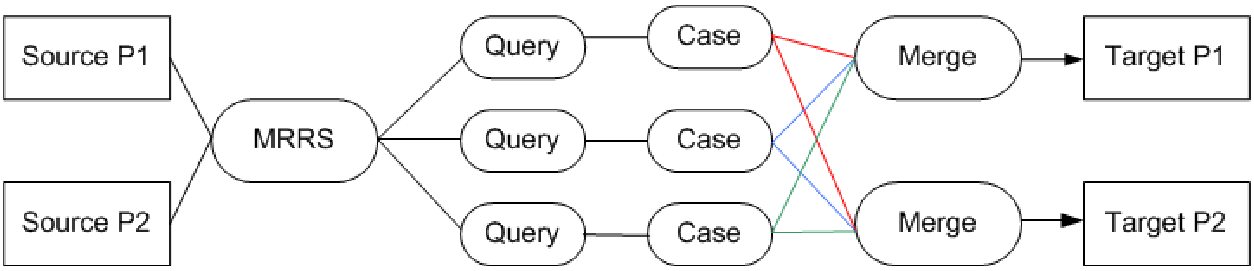
6.7.2.3.2 Two source partitions and a DOP of two

When the number of source partitions is the same as the value for DOP, the engine merges before the target (or before any operation that requires a merge, such as aggregation operations) and proceeds in a single stream to complete the flow.



6.7.2.3.3 Two source partitions, DOP of three, two target partitions

When the number of source partitions is less then the value for DOP, the input feeds into a merge-round-robin splitter (MRRS) that merges the input streams and splits them into a number equal to the value for DOP. The engine then merges the data before the target to equal the number of target partitions, then proceeds to complete the flow.



➜ Tip

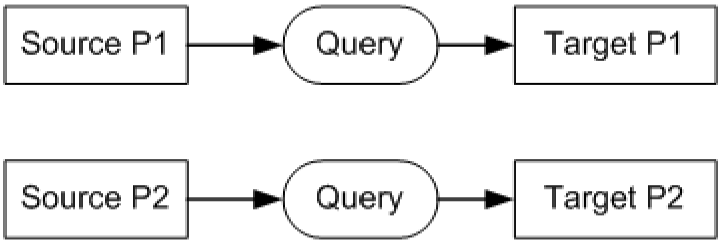
If the number of target partitions is not equal to the number of source partitions, set the *Number of loaders* option equal to the DOP value and do not enable partitioning for the target. Depending on the number of CPUs available, set the DOP value equal to the number of source partitions as a general rule. This produces a data flow without the Merge Round Robin Split and each partition pipes the data directly into the consuming transform.

6.7.2.3.4 Two source partitions, DOP of two, and two target partitions

The best case situation is when the following conditions exist:

- The source and target are partitioned the same way.
- The source and target have the same number of partitions.
- DOP is equal to the same number of partitions.

When a source has two partitions, it replicates twice. Because the DOP value is two, the Query transform replicates twice. When a target has two partitions, it replicates twice. The following figure shows that each source partition feeds directly into a replicated Query transform, and the output from each query feeds directly into a replicated target.



6.7.2.4 File multi-threading

You can set the number of threads used to process some sources and targets. The *Parallel process threads* option is available on the:

- File format editor
- Source file editor
- Target file editor
- Properties window of an ABAP data flow

Without multi-threading:

- With delimited file reading, the SAP Data Services reads a block of data from the file system and then scans each character to determine if it is a column delimiter, a row delimiter, or a text delimiter. Then it builds a row using an internal format.
- For positional file reading, the software does not scan character by character, but it still builds a row using an internal format.
- For file loading, processing involves building a character-based row from the internal row format.

You can set these time-consuming operations to run in parallel. You can use the *Parallel process threads* option to specify how many threads to execute in parallel to process the I/O blocks.

Note

Enabling CPU hyperthreading can negatively affect the performance of servers and is therefore not supported.

Related Information

Designer Guide: File Formats [page 254]
Reference Guide: Objects, Source [page 947]
Reference Guide: Objects, Target [page 960]

6.7.2.4.1 Flat file sources

To use the *Parallel process threads* option, the following conditions must be met:

- In the file format editor:
  - For delimited files, no text delimiters are defined.
    For fixed-width files, having a text delimiter defined does not prevent the file from being read by parallel process threads.
    You can set SAP Data Services to read flat file data in parallel in most cases because the majority of jobs use fixed-width or column-delimited source files that do not have text delimiters specified.
  - An end-of-file (EOF) marker for the file's input/output style is not specified.
  - The value of the row delimiter is not set to `{none}`. A row delimiter can be `{none}` only if the file is a fixed-width file.
  - If the file has a multi-byte locale and you want to take advantage of parallel process threads, set the row delimiter as follows:
    - The length of the row delimiter must be 1. If the codepage of the file is UTF-16, the length of the row delimiter can be 2.
    - The row delimiter hex value must be less than 0x40.
- In the Source File Editor, no number has been entered for *Rows to read*.
  The Rows to read option indicates the maximum number of rows that the software reads. It is normally used for debugging. Its default value is none.
- The maximum row size does not exceed 128 KB.

If a file source needs to read more than one file (for example, *.txt is specified for the *File(s)* option in the file format editor), the software processes the data in the first file before the data in the next file. It performs file multi-threading one file at a time.

6.7.2.4.2 Flat file targets

If you enter a positive value for *Parallel process threads*, Data Services parallel processes flat file targets when the maximum row size does not exceed 128KB.

6.7.2.4.3 Tuning performance

The *Parallel process threads* option is a performance enhancement for some sources and targets. Performance is defined as the total elapsed time used to read a file source.

A multi-threaded file source or target achieves high performance by maximizing the utilization of the CPUs on your Job Server computer. You will notice higher CPU usage when you use this feature. You might also notice higher memory usage because the number of process threads you set (each consisting of blocks of rows that use 128 kilobytes) reside in memory at the same time.

To tune performance, adjust the value for *Parallel process threads*. Ideally, have at least as many CPUs as process threads. For example, if you enter the value 4 for *Parallel process threads*, have at least four CPUs on your Job Server computer.

However, increasing the value for process threads does not necessarily improve performance. The file reads and loads achieve their best performance when the work load is distributed evenly among all the CPUs and the speed of the file's input/output (I/O) thread is comparable with the speed of the process threads.

The I/O thread for a file source reads data from a file and feeds it to process threads. The I/O thread for a file target takes data from process threads and loads it to a file. Therefore, if a source file's I/O thread is too slow to keep the process threads busy, there is no need to increase the number of process threads.

If there is more than one process thread on one CPU, that CPU will need to switch between the threads. There is an overhead incurred in creating these threads and switching the CPU between them.

6.7.2.4.4 Tips

The best value for *Parallel process threads* depends on the complexity of your data flow and the number of available processes. If your Job Server is on a computer with multiple CPUs, the values for file sources and targets should be set to at least two.

After that, experiment with different values to determine the best value for your environment.

Here are some additional guidelines:

- If *Parallel process threads* is set to `none`, then flat file reads and loads are not processed in parallel.
- If *Parallel process threads* is set to `1`, (meaning that one process thread will spawn) and your Job Server computer has one CPU, then reads and loads can occur faster than single-threaded file reads and loads because SAP Data Services reads the I/O thread separately and concurrently with the process thread.
- If *Parallel process threads* is set to `4`, four process threads will spawn. You can run these threads on a single CPU. However, using four CPUs would more likely maximize the performance of flat file reads and loads.

6.8 Distributing Data Flow Execution

The previous section describes how SAP Data Services can run a single process as multiple threads that run in parallel on a multiprocessor computer. Using degree of parallelism (DOP), it can execute each thread on a separate CPU on the computer.

This section describes how the software can split a process (data flow) into multiple processes (sub data flows) that can take advantage of more memory across multiple computers or on the same computer that has more than two gigabytes of memory. For example, if your computer has eight gigabytes of memory, you can have four sub data flows that each can use up to two gigabytes.

With this capability, the software can distribute CPU-intensive and memory-intensive operations (such as join, grouping, table comparison and lookups). This distribution of data flow execution provides the following potential benefits:

- Better memory management by taking advantage of more CPU power and physical memory.
- Better job performance and scalability by taking advantage of grid computing.

You can create sub data flows so that the software does not need to process the entire data flow in memory at one time. You can also distribute the sub data flows to different job servers within a server group to use additional memory and CPU resources.

6.8.1 Splitting a data flow into sub data flows

6.8.1.1 Run as a separate process option

If your data flow contains multiple resource-intensive operations, you can run each operation as a separate process (sub data flow) that uses separate resources (memory and computer) from each other to improve performance and throughput. When you specify multiple *Run as separate process* options in objects in a data flow, SAP Data Services splits the data flow into sub data flows that run in parallel.

The *Run as a separate process* option is available on resource-intensive operations that including the following:

- Hierarchy_Flattening transform
- Associate transform
- Country ID transform
- Global Address Cleanse transform
- Global Suggestion Lists transform
- Match Transform
- United States Regulatory Address Cleanse transform
- User-Defined transform
- Query operations that are CPU-intensive and memory-intensive:
  - Join
  - GROUP BY
  - ORDER BY
  - DISTINCT
- Table_Comparison transform
- Lookup_ext function
- Count_distinct function
- Search_replace function

6.8.1.2 Examples of multiple processes for a data flow

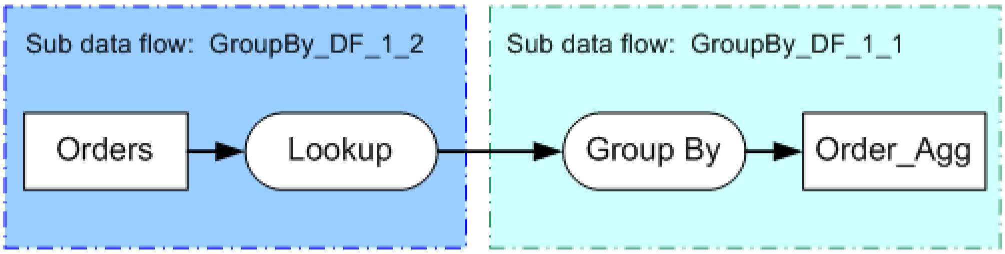
A data flow can contain multiple resource-intensive operations that each require large amounts of memory or CPU utilization. You can run each resource-intensive operation as a separate process that can use more memory on a different computer or on the same computer that has more than two gigabytes of memory. For example, you might have a data flow that sums sales amounts from a lookup table and groups the sales by country and region to find which regions are generating the most revenue. Other than the source and target, the data flow contains a Query transform for the lookup_ext function to obtains sales subtotals and another Query transform to group the results by country and region.

To define separate processes in this sample data flow, take one of the following actions:

- When you define the Lookup_ext function in the first query transform, select the *Run as a separate process* option.
- When you define the Group By operation in the second query transform, select the *Run GROUP BY as a separate process* option on the *Advanced* tab.

6.8.1.2.1 Scenario 1: Run multiple sub data flows with DOP set to 1

The following diagram shows how SAP Data Services splits this data flow into two sub data flows when you specify the *Run as a separate process* option for either the Lookup_ext function or the Group By.



The software generates sub data flow names that follow this format:

```
<DFName_executionGroupNumber_indexInExecutionGroup >
```

- `<DFName>` is the name of the data flow.
- `<executionGroupNumber>` is the order that the software executes a group of sub data flows
- `<indexInExecutionGroup>` is the sub data flow within an execution group.

When you execute the job, the trace log shows two sub data flows that execute in parallel and have different process IDs (Pids). For example, the following depiction of the trace log shows two sub data flows GroupBy_DF_1_1 and GroupBy_DF_1_2 that each start at the same time and have different Pids than the parent data flow GroupBy_DF.

Pid	Tid	Type	Message
...	...	...	...
964	5128	DATAFLOW	Process to execute data flow <GroupBy_DF> is started.
...	...	...	...
4512	5876	DATAFLOW	Process to execute sub data flow <GroupBy_DF_1_1> is started.

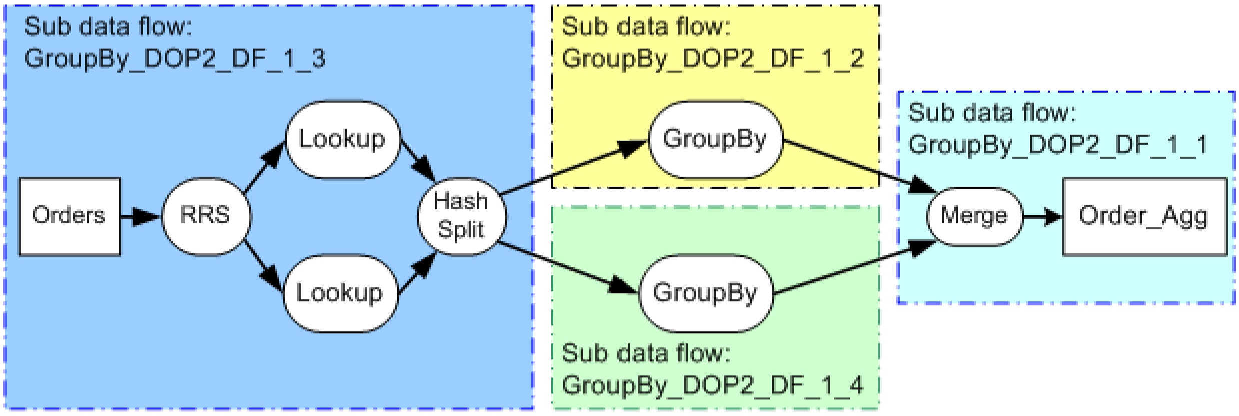
Pid	Tid	Type	Message
396	6128	DATAFLOW	Process to execute sub data flow <GroupBy_DF_1_2> is started.
...	...	...	...

6.8.1.2.2 Scenario 2: Run multiple sub data flows with DOP greater than 1

When the degree of parallelism (DOP) is set to a value greater than 1, each transform defined in the data flow replicates for use on a parallel subset of data.

Set DOP to a value greater than 1 on the data flow Properties window.

The following diagram shows the sub data flows that Data Services generates for GroupBy_DOP2_Job when the *Run GROUP BY as a separate process* is selected and DOP set to 2.



When you execute the job, the trace log shows that the software creates sub data flows that execute in parallel with different process IDs (Pids). For example, the following table shows what the trace log would display for the following four sub data flows that start concurrently, each with a different Pid than the parent data flow GroupBy_DOP2_DF:

- GroupBy_DOP2_DF_1_1
- GroupBy_DOP2_DF_1_2
- GroupBy_DOP2_DF_1_3
- GroupBy_DOP2_DF_1_4

Pid	Tid	Type	Message
...	...	...	...
4288	1960	DATAFLOW	Process to execute data flow <GroupBy_DOP2_DF> is started.

Pid	Tid	Type	Message
...	...	...	...
5548	3636	DATAFLOW	Process to execute sub data flow <GroupBy_DOP2_DF_1_1> is started.
4032	2868	DATAFLOW	Process to execute sub data flow <GroupBy_DOP2_DF_1_2> is started.
1800	5848	DATAFLOW	Process to execute sub data flow <GroupBy_DOP2_DF_1_3> is started.
4416	6128	DATAFLOW	Process to execute sub data flow <GroupBy_DOP2_DF_1_4> is started.
...	...	...	...

➜ Tip

When your data flow has a DOP of greater than one, select either `job` or `data flow` for the *Distribution level* option when you execute the job. If you execute the job with the value `sub data flow` for *Distribution level*, the Round-Robin Split or Hash Split sends data to the replicated queries that might be executing on different job servers. Because the data is sent on the network between different job servers, the entire data flow might be slower.

Related Information

Degree of parallelism [page 2142]

Using grid computing to distribute data flow execution [page 2160]

6.8.1.3 Data_Transfer transform

The Data_Transfer transform creates transfer tables in datastores to enable the software to push down operations to the database server. The Data_Transfer transform creates two sub data flows and uses the transfer table to distribute the data from one sub data flow to the other sub data flow. The sub data flows execute serially.

Related Information

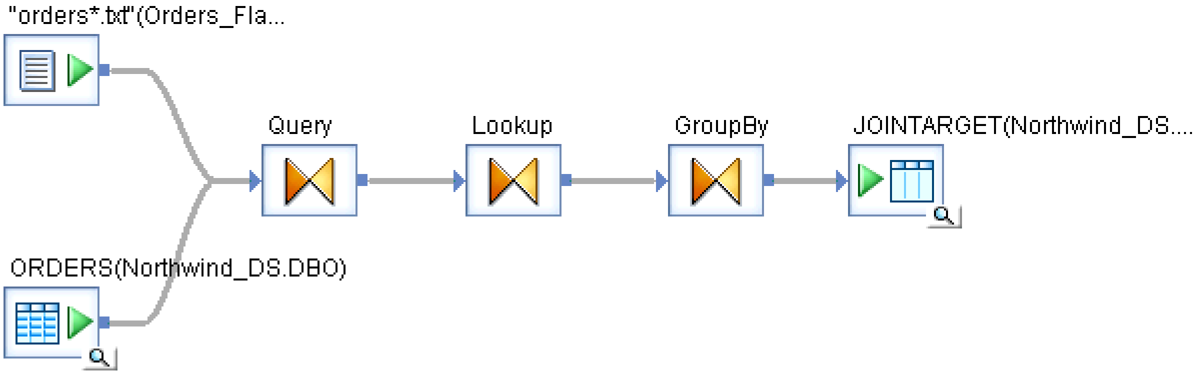
Reference Guide: Transforms, Data_Transfer [page 1071]

6.8.1.4 Examples of multiple processes with Data_Transfer

The following are typical scenarios of when you might use the Data_Transfer transform to split a data flow into sub data flows to push down operations to the database server.

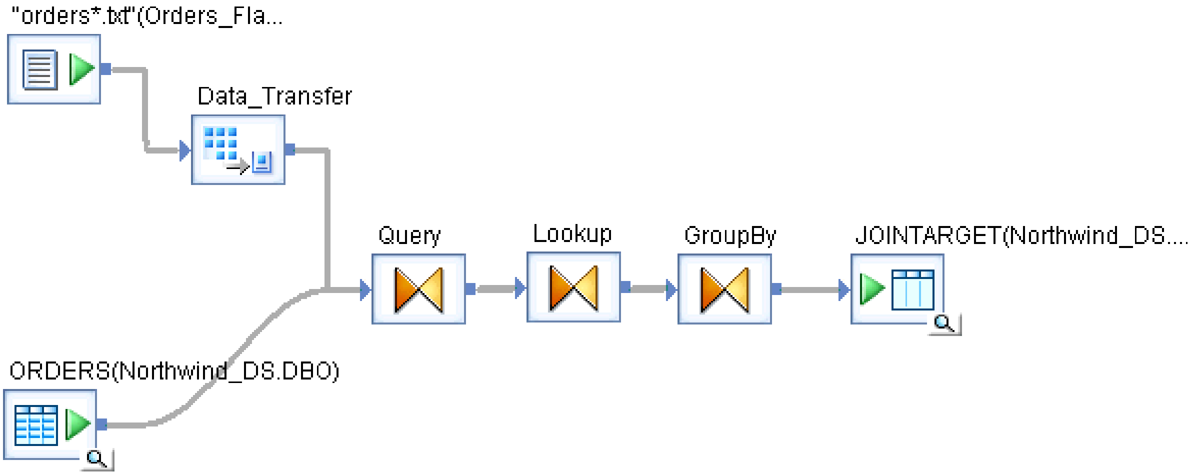
6.8.1.4.1 Scenario 1: Sub data flow to push down join of file and table sources

Your data flow might join an Orders flat file and a Orders table, perform a lookup_ext function to obtain sales subtotals, and another Query transform to group the results by country and region.



6.8.1.4.1.1 Define sub data flows to push down a join of a file and table

1. Add a Data_Transfer transform between the Orders file source and the Query transform.



2. Select the value `Table` from the drop-down list in the *Transfer type* option in the Data_Transfer editor.
3. For *Table name* in the Table options area, browse to the datastore that contains the source table that the Query joins to this file. Double-click the datastore name and enter a name for the transfer table on the Input table for Data_Transfer window.

   In this example, browse to the same datastore that contains the Orders table and enter Orders_FromFile in Table name.
4. After you save the data flow and click *ValidationDisplay Optimized SQL...*, the Optimized SQL window shows that the join between the transfer table and source Orders table is pushed down to the database.

   ```
   SELECT "Data_Transfer_Orders_Flatfile"."PRODUCTID" , "ORDERS"."SHIPCOUNTRY" ,
   "ORDERS"."SHIPREGION" , "Data_Transfer_Orders_Flatfile"."ORDERID"
   FROM "DBO"."ORDERS_FROMFILE"
   "Data_Transfer_Orders_Flatfile","DBO"."ORDERS""ORDERS"
   WHERE ("Data_Transfer_Orders_Flatfile"."ORDERID" = "ORDERS"."ORDERID")
   ```

   SAP Data Services can push down many operations without using the Data_Transfer transform.
5. When you execute the job, the Trace Log shows messages that indicate that the software created two sub data flows with different Pids to run the different operations serially.

Related Information

Push-down operations [page 2116]

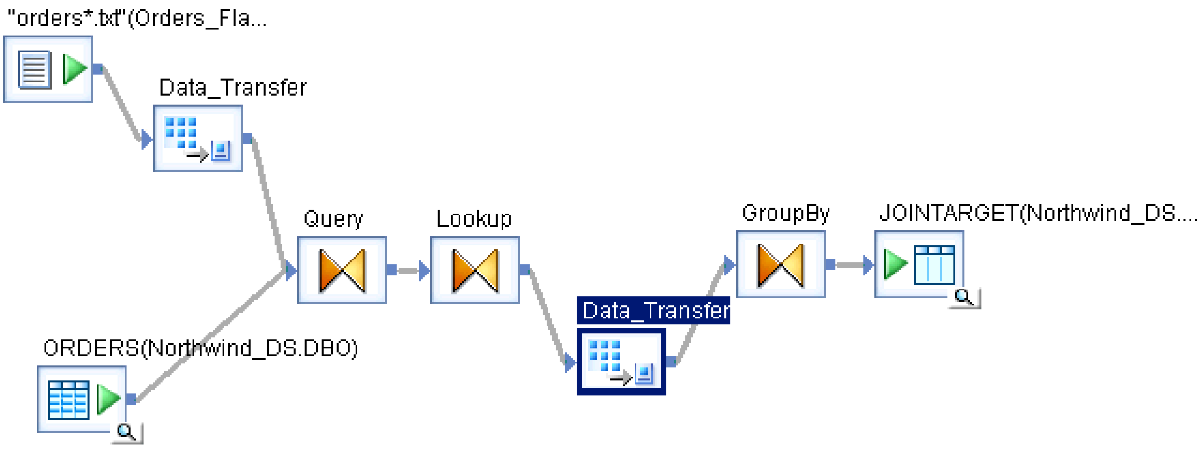
6.8.1.4.2 Scenario 2: Sub data flow to push down memory-intensive operations

You can use the Data_Transfer transform to push down memory-intensive operations such as Group By or Order By.

For the sample data flow in *Scenario 1: Sub data flow to push down join of file and table sources* [page 2157], you might want to push down the Group By operation.

6.8.1.4.2.1 Define sub data flows to push down another operation

1. Add a Data_Transfer transform between the Lookup and GroupBy Query transforms, as the following diagram shows.



2. Select the value `Table` from the drop-down list in the *Transfer type* option in the Data_Transfer editor.
3. For *Table name* in the Table options area, browse to the datastore that contains the target table. Double-click the datastore name and enter a name for the transfer table on the Input table for Data_Transfer window.
4. After you save the data flow and click ▶ *Validation* ▶ *Display Optimized SQL* ◀, the Optimized SQL window shows that the software pushes the GroupBy down to the target database.

```
INSERT INTO "DBO"."JOINTARGET"("PRODUCTID","SHIPCOUNTRY","SHIPREGION","SALES")

SELECT "Data_Transfer_1_Lookup"."PRODUCTID",
"Data_Transform_1_Lookup"."SHIPCOUNTRY",
"Data_Transfer_1_Lookup"."SHIPREGION",sum("Data_Transfer_1_Lookup"."SALES")
FROM "DBO"."GROUPTRANS""Data_Transfer_1_Lookup"
GROUP BY
"Data_Transfer_1_Lookup"."PRODUCTID","Data_Transfer_1_Lookup"."SHIPCOUNTRY",
"Data_Transfer_1_Lookup"."SHIPREGION"
```

   The software can push down many operations without using the Data_Transfer transform.
5. When you execute the job, the messages indicate that the software creates three sub data flows to run the different operations serially.

Related Information

Push-down operations [page 2116]

6.8.2 Using grid computing to distribute data flow execution

SAP Data Services takes advantage of grid computing when you:

- Define a group of Job Servers (called a *Server Group* [page 2160]) that acts as a server grid. The software leverages available CPU and memory on the computers where the Job Servers execute.
- Specify *Distribution levels for data flow execution* [page 2160] to process smaller data sets or fewer transforms on different Job Servers in a Server Group. Each data flow or sub data flow consumes less virtual memory.

6.8.2.1 Server Group

You can distribute the execution of a job or a part of a job across multiple Job Servers within a Server Group to better balance resource-intensive operations. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at runtime.

Related Information

Management Console Guide: Server Groups [page 1885]

6.8.2.2 Distribution levels for data flow execution

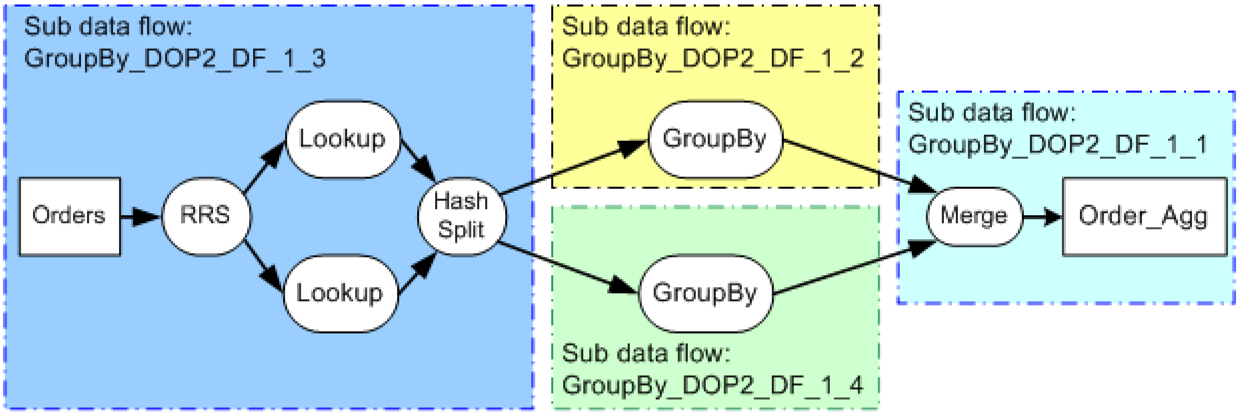
When you execute a job, you can specify the following values on the *Distribution level* option:

- Job level - An entire job can execute on an available Job Server.
- Data flow level - Each data flow within a job can execute on an available Job Server and can take advantage of additional memory (up to two gigabytes) for both in-memory and pageable cache on another computer.
- Sub data flow level - A resource-intensive operation (such as a sort, table comparison, or table lookup) within a data flow can execute on an available Job Server. Each operation can take advantage of up to two gigabytes additional memory for both in-memory and pageable cache on another computer.

6.8.2.2.1 Job level

When you choose a Server Group to execute your job, the default distribution level is Job.

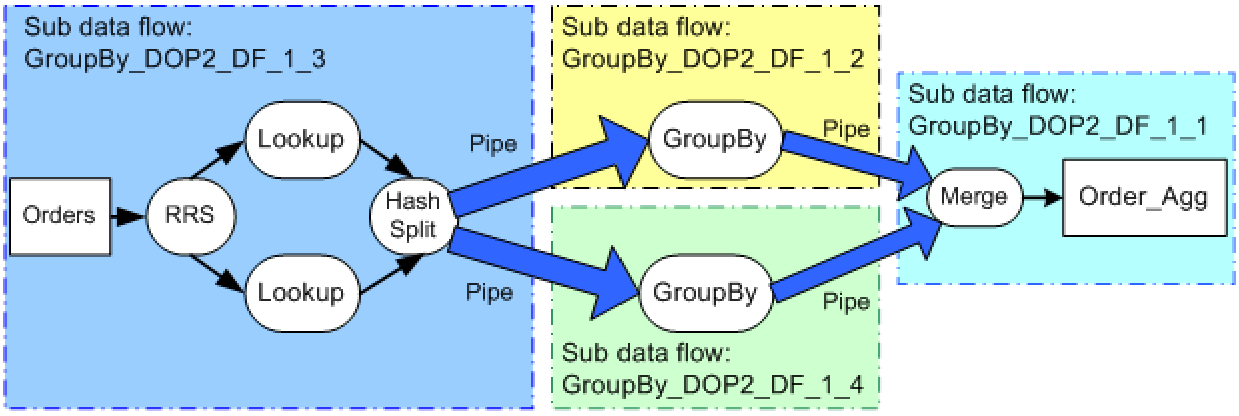
When *Distribution level* has the value `Job`, all of the processes that belong to the job execute on the same computer. For example, section *Scenario 2: Run multiple sub data flows with DOP greater than 1* [page 2155] describes the data flow GroupBy_DOP2_DF which is designed to generate four sub data flows as follows.


Sub data flow: GroupBy_DOP2_DF_1_3
Orders
RRS
Lookup
Lookup
Hash Split
Sub data flow: GroupBy_DOP2_DF_1_2
GroupBy
Sub data flow: GroupBy_DOP2_DF_1_4
GroupBy
Sub data flow: GroupBy_DOP2_DF_1_1
Merge
Order_Agg

When you execute the job, the following Trace log messages indicate the distribution level for each sub data flow:

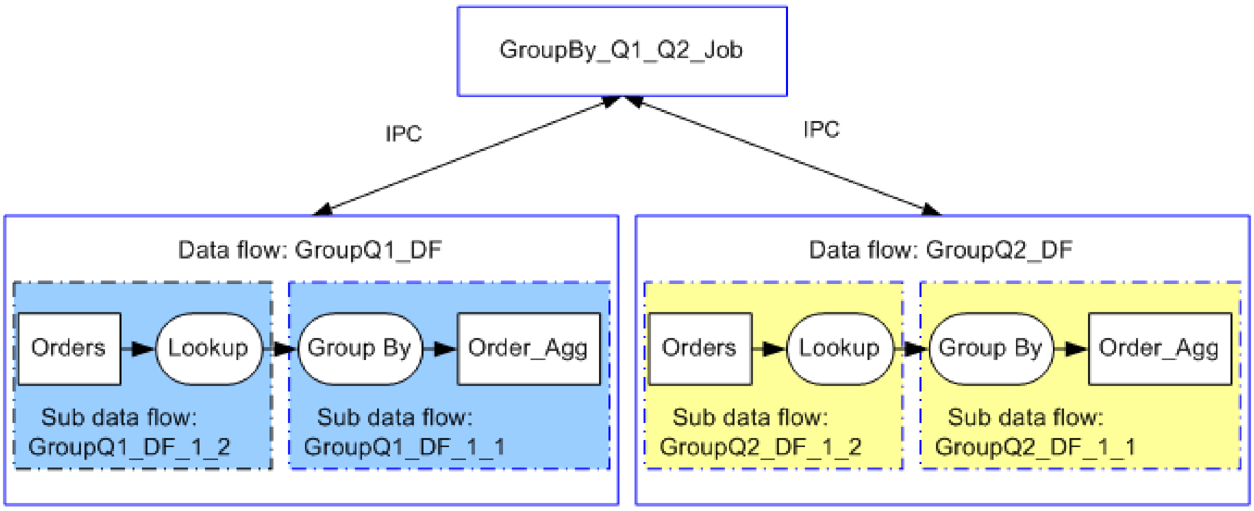
```
Starting sub data flow <GroupBy_DOP2_DF_1_1> on job server host <SJ-C>, port
<3502>. Distribution level <Job>.
Starting sub data flow <GroupBy_DOP2_DF_1_2> on job server host <SJ-C>, port
<3502>. Distribution level <Job>.
Starting sub data flow <GroupBy_DOP2_DF_1_3> on job server host <SJ-C>, port
<3502>. Distribution level <Job>.
Starting sub data flow <GroupBy_DOP2_DF_1_4> on job server host <SJ-C>, port
<3502>. Distribution level <Job>.
```

When *Distribution level* is `Job`, the software uses named pipes to send data between the sub data flow processes on the same computer, as the following diagram indicates with the blue arrows.


Sub data flow: GroupBy_DOP2_DF_1_3
Orders
RRS
Lookup
Lookup
Hash Split
Pipe
Pipe
Sub data flow: GroupBy_DOP2_DF_1_2
GroupBy
Pipe
Sub data flow: GroupBy_DOP2_DF_1_4
GroupBy
Pipe
Sub data flow: GroupBy_DOP2_DF_1_1
Merge
Order_Agg

6.8.2.2.2 Data flow level

When *Distribution level* has the value `Data flow`, all of the processes that belong to each data flow can execute on a different computer. For example, the following GroupBy_Q1_Q2_Job has two data flows: GroupQ1_DF and GroupQ1_DF that process orders for the first quarter and second quarter, respectively.



- The solid blue lines enclose each process that can execute on a separate Job Server. In this example, each data flow can execute on a different computer than the computer where the job started.
- SAP Data Services uses Inter-Process Communications (IPC) to send data between the job and data flows on the different computers. IPC uses the peer-to-peer port numbers specified on the *Start port* and *End port* options in the Server Manager.

Note

The default values for *Start port* and *End port* are `1025` and `32767`, respectively. Change these values if you want to restrict the number of ports or if some of the ports are already in use.

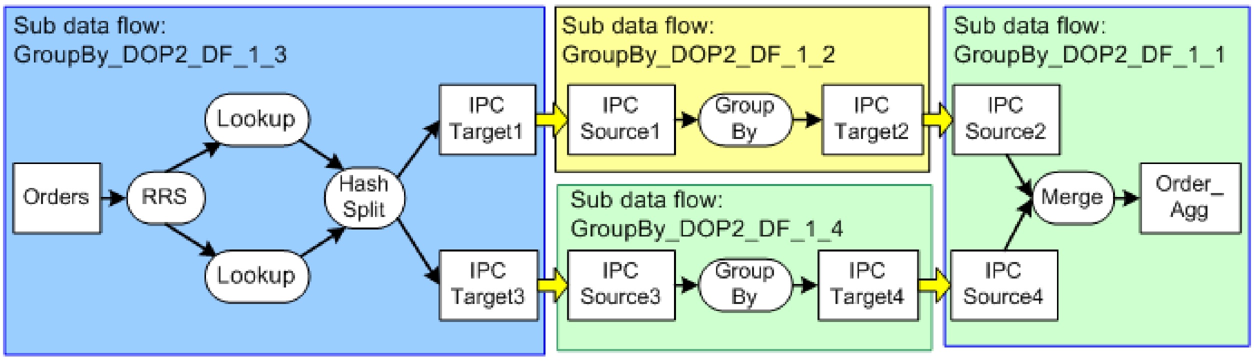
When you execute the job, the Trace log displays messages such as the following that indicate the communication port for the data flow and the distribution level for each data flow. All of the sub data flows within a data flow run on the same computer.

```
Data flow communication using peer-to-peer method with the port range <1025> to
<32767>.
...
Peer-to-peer connection server for session process is listening at host <SJ-C>,
port <1025>.
Job <GroupBy_Q1_Q2_Job> is started.
Starting data flow </GroupBy_Q1_Q2_Job/GroupBy_Q1_DF> on job server host <SJ-C>,
port <3502>. Distribution level <Data
flow>. Data flow submitted to server group <sg_direpo>. Load balancing algorithm
<Least load>. Server group load statistics
from job server <mssql_lap_js SJ-C 3502>:
<mssql_lap_js SJ-C 3502> System Load <47%> Number of CPUs <1>
<MSSQL2005_JS SJ-W-C 3500> System Load <70%> Number of CPUs <2>
Process to execute data flow <GroupBy_Q1_DF> is started.
Starting sub data flow <GroupBy_Q1_DF_1_1> on job server host <SJ-C>, port <3502>.
Distribution level <Data flow>.
Starting sub data flow <GroupBy_Q1_DF_1_2> on job server host <SJ-C>, port <3502>.
Distribution level <Data flow>.
Starting sub data flow <GroupBy_Q1_DF_1_3> on job server host <SJ-C>, port <3502>.
```

```
Distribution level <Data flow>.
Starting sub data flow <GroupBy_Q1_DF_1_4> on job server host <SJ-C>, port <3502>.
Distribution level <Data flow>.
```

6.8.2.2.3 Sub data flow level

When *Distribution level* has the value `Sub data flow`, each sub data flow within a data flow can execute on a different computer. In the example that section *Scenario 2: Run multiple sub data flows with DOP greater than 1* [page 2155] describes, the GroupBy_DOP2_Job has four sub data flows as follows.



- The solid blue lines enclose each process that can execute on a separate Job Server. In this example, each sub data flow can execute on a different computer than the computer where the job started.
- The yellow arrows indicate the Inter-Process Communications (IPC) that SAP Data Services uses to send data between the job and sub data flows on the different computers. IPC the peer-to-peer port numbers specified on the *Start port* and *End port* options in the Server Manager.
  The default values for Start port and End port are 1025 and 32767, respectively. Change these values if you want to restrict the number of ports or if some of the ports are already in use.

> i Note
>
> If you find that sending data across the network is causing your data flow to execute longer, you might want to change *Distribution level* from `Sub data flow` to `Data flow` or `Job`.

When you execute the job, the Trace log displays messages such as the following that indicate that the software selects a job server for each sub data flow based on the system load on each computer:

```
Starting sub data flow <GroupBy_DOP2_DF_1_1> on job server host <SJ-C>, port
<3502>. Distribution level <Sub data flow>. Sub data
flow submitted to server group <sg_direpo>. Load balancing algorithm <Least load>.
Server group load statistics
from job server <mssql_lap_js SJ-C 3502>:
<mssql_lap_js SJ-C 3502> System Load <21%> Number of CPUs <1>
<MSSQL2005_JS SJ-W-C 3500> System Load <70> Number of CPUs <1>
Starting sub data flow <GroupBy_DOP2_DF_1_2> on job server host <SJ-C>, port
<3502>. Distribution level <Sub data
flow>. Sub data flow submitted to server group <sg_direpo>. Load balancing
algorithm <Least load>. Server group load statistics
from job server <mssql_lap_js SJ-C 3502>:
<mssql_lap_js SJ-C 3502> System Load <21%> Number of CPUs <1>
<MSSQL2005_JS SJ-W-C 3500> System Load <70> Number of CPUs <2>
```

The following messages show the communication port that each sub data flow uses:

```
Peer-to-peer connection server for sub data flow <GroupBy_DOP2_DF_1_1> is listening
at host <SJ-C>, port <1027>.
Process to execute sub data flow <GroupBy_DOP2_DF_1_4> is started.
Peer-to-peer connection server for sub data flow <GroupBy_DOP2_DF_1_2> is listening
at host <SJ-C>, port <1028>.
Peer-to-peer connection server for sub data flow <GroupBy_DOP2_DF_1_3> is listening
at host <SJ-C>, port <1029>.
Peer-to-peer connection server for sub data flow <GroupBy_DOP2_DF_1_4> is listening
at host <SJ-C>, port <1030>.
```

6.9 Bulk Loading and Reading

SAP Data Services supports capabilities present in most supported databases that enable you to load and in some cases read data in bulk rather than using SQL statements. Some general considerations when using bulk loading and reading are:

- Specify bulk-loading options on the Data Services target table editor on the *Options* and *Bulk Loader Options* tabs.
- Specify Teradata reading options on the source table editor *Teradata options* tab.
- Most databases do not support bulk loading with a template table.

For details on the options for each database type, see the *Reference Guide*.

Related Information

Reference Guide: Objects, Target tables [page 963]

Reference Guide: Objects, Teradata source [page 952]

6.9.1 Bulk loading in DB2 Universal Database

SAP Data Services supports bulk loading to the DB2 Universal Database.

6.9.1.1 When to use each DB2 bulk-loading method

SAP Data Services supports multiple bulk-loading methods for DB2 Universal Database (UDB) on Windows and UNIX. The following table lists the methods that you can select depending on your requirements.

Note

You cannot bulk load data to DB2 databases that run on AS/400 or z/OS (MVS) systems.

Load method	Description	Advantages	Restrictions
CLI Load	Loads a large volume of data at high speed by passing it directly from memory to the table on the DB2 UDB server.	• Provides the fastest way to bulk load. • Eliminates some parameters because no intermediate data file is required. • Can put rows that violate the unique key constraint into an exception table.	• Must specify *Recoverable* and *Copy target directory* options to enable recovery because DB2 logging is not enabled for CLI Load. • The DB2 UDB server and client must be Version 8.0 or later. • Stops loading when it encounters the first rejected row.
Import	Loads a large volume of data by using a SQL INSERT statement to write data from an input file into a table or view.	• Recovery is enabled automatically because DB2 logging occurs during import. • Performs referential integrity or table constraint checking in addition to unique key constraint checking.	• Because DB2 logs each INSERT statement, this method is the slowest way to bulk load data. • The Data Services Job Server and DB2 UDB server must be on the same computer.

6.9.1.2 Using the DB2 CLI load method

The DB2 Call Level Interface (CLI) load method performs faster than the bulk load or import utilities because it does not write the data to an intermediate file. Instead, the CLI load method writes the data from memory (where SAP Data Services extracted or transformed the data) directly to the table on the DB2 server.

6.9.1.2.1 Configuring your system to use the CLI load method

1. Enter the appropriate information in the datastore editor, on the *DB2 Properties* tab.

   Fields include:

   ○ *Bulk loader user name*: The user name SAP Data Services uses when loading data with the CLI load option. For bulk loading, you might specify a different user name, for example one with import and load permissions.

   ○ *Bulk loader password*: The password SAP Data Services uses when loading with the CLI load option.
2. To use a different bulk loader working directory than the default (`<DS_COMMON_DIR>\log\bulkloader`), specify the directory name in the datastore editor on the *Connections* tab.

6.9.1.2.2 Using the CLI load method in a job

1. Open the DB2 target table editor in the Designer workspace.
2. Select the *Bulk Loader Options* tab below the table schema area.
3. In the *Bulk loader* list, select *CLI load*.

   The window updates to show all CLI load options. CLI load options include these existing bulk load options:

   - Mode
   - Warning row count
   - Exception table name
   - Recoverable
   - Copy target directory

Additional or changed CLI load options include:

- *Maximum bind array*: Defines the maximum number of rows extracted or transformed before the software sends the data to the DB2 table or view. If you do not enter a value, Data Services uses the CLI load default value of 10000 rows.
- *Clean up bulk loader directory after load*: If you select this option, the software deletes the message file when the CLI load completes successfully. Because the CLI load obtains the data from memory, Data Services creates no control or data files.

Related Information

Reference Guide: Objects, Target tables [page 963]

6.9.1.3 Using the import utility

SAP Data Services also supports bulk loading in the DB2 Universal Database 5.2 environment using the import utility. For the software to initiate DB2 bulk loading by this method directly, the Job Server and DB2 must be located on the same system. If they are not, use the following procedure to initiate bulk loading:

1. Generate a control file and data file. Check *Generate files only* in the target table editor on the Bulk Loader Options tab.
2. Manually move the control file and data file to the system where the target database is located.
3. Start the execution of the bulk loader manually.

6.9.2 Bulk loading in HP Neoview

SAP Data Services supports bulk loading to HP Neoview via Neoview Transporter.

For detailed information about HP Neoview loading options and their behavior, see the relevant HP Neoview product documentation.

To use Neoview Transporter, you must also install the following components:

- Neoview Transporter Java Client
- Java JRE version 1.5 or newer
- Neoview JDBC Type 4 Driver
- Neoview ODBC Windows Driver (for Windows)
- Neoview UNIX Drivers (for connecting to a database on UNIX)
- Neoview Command Interface

Note

If you are using multibyte data on Windows, you must change the Windows regional settings to the multibyte language, for example, Japanese.

Note

When you install the Java Client, an environment variable called NVTHOME is created and will point to the location of the Neoview Transporter base directory. You may receive an error in SAP Data Services if NVTHOME is not defined.

HP Neoview recommends that you use the bulk-loading method to load data for faster performance. The SAP Data Services bulk loader for HP Neoview supports UPDATE and UPSERT as well as INSERT operations, which allows for more flexibility and performance.

SAP Data Services generates a control file as input into Neoview Transporter. The control file specifies the data files and the target tables to be loaded. Being in UTF-8, the control file supports multibyte data.

By default, HP Neoview uses the SQL insert operation. For SQL update and upsert options, the control file specifies the columns used in the WHERE clause and the columns to be updated in the UPDATE clause. By default, SAP Data Services uses the primary key columns in the WHERE clause.

To bulk load to a HP Neoview target table, the software:

- Creates a control file in UTF-8
- Loads data from the source into the file or named pipe in UTF-8
- Invokes Neoview Transporter

6.9.2.1 How Data Services and HP Neoview use the file options to load

You can choose to use either named pipes or data files as staging for loading the data. Choose from the following file options:

- Data file (for Windows and UNIX)
- Named pipe (for UNIX only)

Data file

SAP Data Services runs bulk-loading jobs using a staging data file as follows:

1. The software generates staging data file(s) containing data to be loaded into a HP Neoview table.
2. The software generates a control file to be used by Neoview Transporter.

Named pipe

SAP Data Services runs bulk-loading jobs using named pipes as follows:

1. The software generates a control file that Neoview Transporter uses to manipulate the database.
2. The software creates a pipe to contain the data to apply into an HP Neoview table.
   On UNIX, the pipe is a FIFO (first in, first out) file that has name of this format:

   ```
   /temp/<filename>.dat
   ```
3. The software invokes Neoview Transporter with the control file as input.
4. The software writes data to the pipes.
5. Neoview Transporter reads data from the pip and applies data to the HP Neoview table.

6.9.2.2 Using the UPSERT bulk operation

The purpose of the HP Neoview Upsert operation is to update a row, but if no row matches the update, the row is inserted.

In SAP Data Services, you enable Upsert on the target table editor's *Bulk Loader Options* tab. In the *Data Services options* section, for *Bulk Operation*, select *Upsert* (the default is *Insert*) in the SQL Operation list.

After selecting *Upsert*, notice you can also enable the *Use input keys* option on the target editor's *Options* tab. The *Use input keys option* will assign the input primary keys as primary keys in the target table.

Related Information

Reference Guide: Objects, Target tables [page 963]

6.9.3 Bulk loading in Informix

SAP Data Services supports Informix bulk loading. For detailed information about Informix bulk-loading utility options and their behavior in the Informix DBMS environment, see the relevant Informix product documentation.

Setting up Informix for bulk-loading requires that you set the `INFORMIXDIR`, `INFORMIXSERVER`, and `PATH` environment variables.

For the software to initiate Informix bulk loading directly, the Job Server and the target database must be located on the same system.

> Note
>
> SAP Data Services provides Informix bulk-loading support only for single-byte character ASCII delimited files (not for fixed-width files).

6.9.3.1 Setting Informix server variables

For Windows platforms, configure the environment variables in the `$LINK_DIR\bin\dbloadIfmx.bat` script.

```
set INFORMIXDIR=<C:\path\to\informix\installation>
set INFORMIXSERVER=ol_svr_custom
set PATH=%INFORMIXDIR%\bin;%PATH%
```

For UNIX platforms, configure the environment variables in the `$LINK_DIR/bin/dbloadIfmx.sh` script.

```
export INFORMIXDIR=</path/to/informix/installation>
export INFORMIXSERVER=ol_svr_custom
export PATH=$INFORMIXDIR/bin:$PATH
```

6.9.4 Bulk loading in Microsoft SQL Server

SAP Data Services supports Microsoft SQL Server bulk loading through the SQL Server ODBC bulk copy API. For detailed information about the SQL Server ODBC bulk copy API options and their behavior in the Microsoft SQL Server DBMS environment, see the relevant Microsoft SQL Server product documentation.

6.9.4.1 To use the SQL Server ODBC bulk copy API

To enable bulk loading on the default job server, do the following:

1. From the *Tools* menu, select *Options* > *Job Server* > *General*.
2. For *Section*, enter `AL_Engine`.
3. For *Key*, enter `UseSQLServerBulkCopy`.
4. Select TRUE (default) or FALSE. If you leave the default, the software uses the SQL Server ODBC bulk copy API. If you set this parameter to FALSE, the software overrides the default and uses the SQLBulkOperations API (this method is slower than the default).

6.9.4.2 Network packet size option

When loading to SQL Server, the client caches rows until it either fills a network packet or reaches the commit size (regardless of whether the packet is full). Then the client sends the packet to the server. You can affect performance by tuning commit size and network packet size. You can change these sizes on the Bulk Loader Options tab for SQL Server:

- Rows per commit
  This option lets you specify the number of rows to put in the cache before issuing a commit.
- Network packet size
  This option lets you specify network packet size in kilobytes. The default packet size is 4 kilobytes.

Note

It is recommended that you set the *Rows per commit* and *Network packet size* parameters to avoid sending many partially filled packets over the network and ensure that the packet size contains all rows in the commit.

6.9.4.3 Maximum rejects option

The *Maximum rejects* parameter (on the Bulk Loader Options page) can also affect your SQL Server bulk-loading performance. When you set *Maximum rejects* to 0, SAP Data Services stops at the first error it encounters and does not cache rows in the transaction (caching rows in a transaction allows the software to process each row even if an error occurs during the transaction commit process.)

When you do not specify a value for *Maximum rejects*, the software ignores the rejected rows, logs warnings, and continues processing.

6.9.5 Bulk loading in Netezza

SAP Data Services supports bulk loading to Netezza Performance Servers.

For detailed information about Netezza loading options and their behavior in the Netezza environment, see the relevant Netezza product documentation.

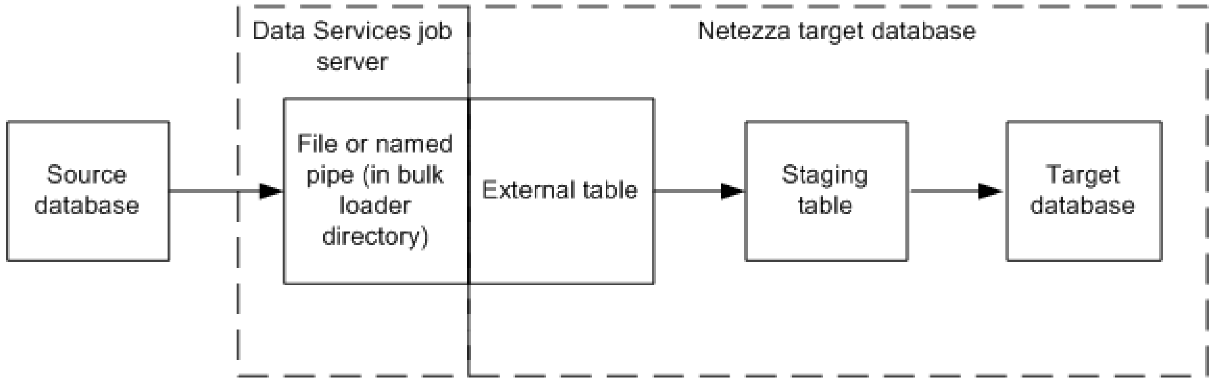
Netezza recommends using the bulk-loading method to load data for faster performance. Unlike some other bulk loaders, the SAP Data Services bulk loader for Netezza supports UPDATE and DELETE as well as INSERT operations, which allows for more flexibility and performance.

6.9.5.1 Netezza bulk-loading process

To bulk load to a Netezza target table, SAP Data Services:

- Creates an external table that is associated with a local file or named pipe.
- Loads data from the source into the file or named pipe.

- Loads data from the external table into a staging table by executing an INSERT statement.
- Loads data from the staging table to the target table by executing a set of INSERT/UPDATE/DELETE statements.



6.9.5.2 Options overview

From the *Bulk Loader Options* tab of your Netezza target table editor, select one of these methods depending on your Netezza environment:

- *Named pipe*—SAP Data Services streams data as it is written to the named pipe through the external table to the staging table. For files that are larger than 4 GB in size, select this option for faster performance.
- *File*—SAP Data Services writes the data to a file before loading through the external table to the staging table. For files that are smaller than 4 GB in size, select this option for faster performance.
- *None*—SAP Data Services does not use bulk loading.

Because the bulk loader for Netezza also supports UPDATE and DELETE operations, the following options (on the target table editor *Options* tab) are also available for Netezza bulk loading.

- Column comparison
- Number of loaders
- Use input keys
- Update key columns
- Auto correct load

Related Information

Reference Guide: Objects, Target tables [page 963]

6.9.5.3 Configuring bulk loading for Netezza

First configure the bulk loader and log directories in the datastore editor, then enable and configure bulk loading in the target table editor:

1. In the Netezza datastore editor, click *Advanced*.
2. Click in the field to the right of *Bulk loader directory* and type the directory path or click *Browse* to where the software should write SQL and data files for bulk loading.
3. In the *FTP* category, enter the FTP host name, login user name, login password, and host working directory.

   These options are used to transfer the Netezza nzlog and nzbad files.

   > Note
   >
   > If this datastore is not being used specifically for Netezza bulk loading, the software ignores any FTP option entries.

4. If you are loading non-ASCII character data, set the *Code page* to *latin-9*.

   If you are loading mulitbyte data, set the *Code page* to *utf-8*.
5. Click *OK* or *Apply*.
6. Open the data flow and open the target table editor by clicking its name.
7. On the *Bulk Loader Options* tab, select a bulk-loading method and configure the remaining options there and on the *Options* tab.
8. Save the data flow.

Related Information

Reference Guide: Objects, Database datastores (ODBC) [page 885]

Designer Guide: Datastores, Defining a database datastore [page 213]

Reference Guide: Objects, Target tables [page 963]

6.9.5.4 Netezza log files

When writing from the external table to the staging table, Netezza generates logs (nzlog and nzbad files) and writes them to a database server working directory. You can use these logs to troubleshoot your jobs. (If you do not enter a *Database server working directory* in the datastore editor, Netezza uses the temp directory on its server, /tmp, to store the nzlog and nzbad files.)

For SAP Data Services to access and manage these logs, configure the FTP parameters in the datastore editor. After a load, trhe software copies these files from the specified Netezza *Database server working directory* to the specified *Bulk loader directory* and deletes them from the Netezza server.

For successful loads, SAP Data Services then deletes these log files from the *Bulk loader directory* (assuming the *Clean up bulk loader directory after load* option is checked in the target table editor).

For failed loads, the software does not delete the log files from the *Bulk loader directory* even if the *Clean up bulk loader directory after load* option is checked in the target table editor.

6.9.6 Bulk loading in Oracle

SAP Data Services supports Oracle bulk loading.

6.9.6.1 Bulk-loading methods

You can bulk load to Oracle using an API or a staging file:

- If you select the *API* method, SAP Data Services accesses the direct path engine of Oracle's database server associated with the target table and connected to the target database. Using Oracle's Direct-Path Load API, input data feeds directly into database files. To use this option, you must have Oracle version 8.1 or later.
- If you select the *File* method, Data Services writes an intermediate staging file, control file, and log files to the local disk and invokes the Oracle SQL*Loader. This method requires more processing time than the API method.
  For detailed information about the Oracle SQL*Loader options, see the relevant Oracle product documentation.

6.9.6.2 Bulk-loading modes

Bulk loading in Oracle supports two modes of data loading: conventional-path and direct-path. Conventional-path loading is implicit for the *File* option if you do not select *Direct-path* on the *Bulk Loader Options* tab in the target table editor. SAP Data Services always uses direct-path loading for the *API* option.

- Conventional-path loading: Conventional-path loads use the SQL INSERT statements to load data to tables.
- Direct-path loading:Direct-path loads use multiple buffers for a number of formatted blocks that load data directly to database files associated with tables.

6.9.6.3 Bulk-loading parallel-execution options

Parallel-execution options for bulk loading are on the *Options* tab.

For the API method, you can choose to select the *Enable partitioning* check box. If selected, SAP Data Services generates the number of target parallel instances based on the number of partitions in the target table. If not selected or if your table target is not partitioned, Data Services uses one loader by default.

For the File method, enter a value in the *Number of loaders* box or select the *Enable partitioning* check box.

> Note
>
> The *Enable partitioning* check box does not appear on the *Options* tab if the target table is not partitioned.

6.9.6.4 Bulk-loading scenarios

With two bulk-loading methods, two load modes, and two parallel load options, there are several scenarios you can configure:

Scenario	Method	Load Mode	Parallel Load Options
1	API	Direct-path	Enable partitions is not selected (One loader is used by default)
2	API	Direct-path	Enable partitions is selected
3	File	Direct-path	Number of loaders = 1
4	File	Direct-path	Number of loaders > 1
5	File	Direct-path	Enable partitions is selected
6	File	Conventional	Number of loaders = 1
7	File	Conventional	Number of loaders > 1
8	File	Conventional	Enable partitions is selected

Here are some tips for using these scenarios:

- The API method always uses the direct-path load type, and when it is used with a partitioned target table, SAP Data Services processes loads in parallel. The software instantiates multiple loaders based on the number of partitions in a table. Each loader receives rows that meet the conditions specified by the partition.
- With the File method, direct-path is faster than conventional load, but the File method is slower than using an API because of the need to generate a staging file, logs, and invoke Oracle's SQL*Loader.
- With the File method, when you use a value of greater than one for either the *Number of Loaders* or the *Enable partitioning* option, loads cannot truly run in parallel. The creation of a staging file and log for each loader is serialized.

6.9.6.5 Using bulk-loading options

As seen in the table on the previous page, there are many ways to set up bulk loading for an Oracle database. The following sections describe two scenarios in detail.

6.9.6.5.1 Direct-path loads using Number of Loaders and File method

In the *Options* tab of the target table editor, when you enter a value for *Number of loaders*, SAP Data Services instantiates multiple loaders. Each loader receives rows equal to the amount specified in the *Rows per commit* box on the *Bulk Loader Options* tab. The loaders pipe rows to a staging file, then call the SQL*Loader to load the staging file contents into the table.

This process occurs in "round-robin" fashion. For example, if you set *Rows per commit* to `5000` and *Number of loaders* to `2`, then the first loader receives 5000 rows, writes them to the staging file, and then invokes the SQL*Loader to load the data into the table.

Meanwhile, the second loader receives the second batch of 5000 rows, writes them to a staging file, and then waits for the first loader to complete the loading. When the first loader completes the bulk load, the second loader starts, and while the second loader is loading, the first loader receives the third batch of 5000 rows. This process continues until all the data loads.

The SQL*Loader uses a control file to read staging files and load data. The software either creates this control file at runtime or uses one that is specified on the *Bulk Loader Options* tab at design time.

For parallel loading, the generated control files, data files, and log files are named as follows:

```
<TableNameTIDPID_LDNUM_BATCHNUM >
```

Where:

<TableName:> The name of the table into which data loads.

<TID> : The thread ID.

<PID> : The process ID.

<LDNUM> : The loader number, which ranges from `0` to number of loaders minus 1. For single loaders, `LDNUM` is always `0`.

<BATCHNUM> : The batch number the loader is processing. For single loaders the **<BATCHNUM>** is always `0`.

> Note
>
> Product performance during this type of parallel loading depends on a number of factors such as distribution of incoming data and underlying DBMS capabilities. Under some circumstances it is possible that specifying parallel loaders can be detrimental to performance. Always test the parallel loading process before moving to production.

6.9.6.5.2 Direct-path loads using partitioned tables and API method

You can import partitioned tables as SAP Data Services metadata.

In the *Options* tab of the target table editor, when you select *Enable partitioning*, the software instantiates multiple loaders based on the number of partitions in a table. Each loader receives rows that meet the conditions specified by the partition. In addition, commits occur based on the number specified in the *Rows per commit* box on the *Bulk Loader Options* tab.

For example:

- If you *Rows per commit* to 5000, the number of partitions is set `2`, and your first partition includes 2500 rows, then the first loader commits after receiving all possible rows (2500) while concurrently processing the second loader.
- If you *Rows per commit* to 5000, the number of partitions is set `2`, and your first partition includes 10,000 rows, then the first loader commits twice. Once after receiving 5000 rows and again after receiving the second batch of 5000 rows. Meanwhile, the second loader is processing its rows.

The loaders pipe rows directly to Oracle database files by using Oracle direct-path load APIs (installed with the Oracle client) that are associated with the target database.

The API method allows the software to bypass the use of the SQL* Loader (and the control and staging files it needs). In addition, by using table partitioning, bulk loaders can pass data to different partitions in the same target table at the same time. Using the API method with partitioned tables fully optimizes performance.

> Note
>
> If you plan to use a partitioned table as a target, the physical table partitions in the database must match the metadata table partitions in SAP Data Services. If there is a mismatch, Data Services will not use the partition name to load partitions, which impacts processing time.

For the API method, the software records and displays error and trace logs as it does for any job. A monitor log records connection activity between components; however, it does not record activity while the API is handling the data.

6.9.7 Bulk loading in SAP HANA

SAP Data Services supports bulk loading to the SAP HANA database.

For improved performance when using changed-data capture or auto correct load, Data Services uses a temporary staging table to load the target table. Data Services first loads the data to the staging table, then it applies the operation codes (INSERT, UPDATE, and DELETE) to update the target table. With the *Bulk load* option selected in the target table editor, any one of the following conditions triggers the staging mechanism:

- The data flow contains a Map_CDC_Operation transform.
- The data flow contains a Map_Operation transform that outputs UPDATE or DELETE rows.
- The data flow contains a Table_Comparison transform.
- The *Auto correct load* option in the target table editor is set to *Yes*.

If none of these conditions are met, that means the input data contains only INSERT rows. Therefore Data Services does only a bulk insert operation, which does not require a staging table or the need to execute any additional SQL.

By default, Data Services automatically detects the SAP HANA target table type and updates the table accordingly for optimal performance.

Because the bulk loader for SAP HANA is scalable and supports UPDATE and DELETE operations, the following options (target table editor ▶ *Options* ▶ *Advanced* ▶ *Update control* ◀) are also available for bulk loading:

- Use input keys
- Auto correct load

Related Information

Reference Guide: Objects, SAP HANA target table options [page 986]

6.9.8 Bulk loading in SAP Sybase ASE

SAP Data Services supports bulk loading of SAP Sybase ASE databases through the SAP Sybase ASE bulk copy utility. For detailed information about the SAP Sybase ASE bulk loader options and their behavior in the SAP Sybase ASE DBMS environment, see the relevant SAP Sybase ASE product documentation.

6.9.9 Bulk loading in SAP Sybase IQ

SAP Data Services supports bulk loading to SAP Sybase IQ databases via the SAP Sybase IQ LOAD TABLE SQL command. For detailed information about the SAP Sybase IQ LOAD TABLE parameters and their behavior in the SAP Sybase IQ database environment, see the relevant SAP Sybase IQ product documentation.

For improved performance when using changed-data capture or auto correct load, Data Services uses a termporary staging table to load the target table. Data Services first loads the data to the staging table, then it applies the operation codes (INSERT, UPDATE, and DELETE) to update the target table. With the *Bulk load* option selected in the target table editor, any one of the following conditions triggers the staging mechanism:

- The data flow contains a Map_CDC_Operation transform.
- The data flow contains a Map_Operation transform that outputs UPDATE or DELETE rows.
- The *Auto correct load* option in the target table editor is set to *Yes*.

If none of these conditions are met, that means the input data contains only INSERT rows. Therefore, Data Services does only a bulk INSERT operation, which does not require a staging table or the need to execute any additional SQL.

Note that because the bulk loader for SAP Sybase IQ also supports UPDATE and DELETE operations, the following options (target table editor *Options* > *Advanced* > *Update control*) are also available for bulk loading:

- Use input keys
- Auto correct load

Related Information

Reference Guide: Objects, SAP Sybase IQ target table options [page 988]

6.9.9.1 Configuring bulk loading for SAP Sybase IQ

First configure the bulk loader and log directories in the datastore editor, then enable and configure bulk loading in the target table editor:

1. In the SAP Sybase IQ datastore editor, click *Advanced*.
2. Click in the field next to *Bulk loader directory* and type the directory path or click *Browse* to navigate to where Data Services should write command and data files for bulk loading.
3. Depending on the version of SAP Sybase IQ to which this datastore connects, configure *Bulk Loader* and/or *FTP* options.
4. Click *OK* or *Apply*.
5. Open the data flow and open the target table editor by clicking its name.
6. On the *Bulk Loader Options* tab, select a bulk-loading method and configure the remaining options there and on the *Options* tab.
7. Save the data flow.

Related Information

Reference Guide: Objects, Database datastores, SAP Sybase IQ [page 898]
Reference Guide: Objects, Target tables, SAP Sybase IQ target table options [page 988]

6.9.9.2 SAP Sybase IQ log files

After a job executes, Data Services stores the SAP Sybase IQ message and row logs in the *Bulk loader directory* specified in the datastore editor (regardless of the setting for the *JS and DB on same machine* option). A data file will also be present if you do not use the named pipe option. If you do not specify a Bulk loader directory, Data Services by default writes the files to the directory `<DS_COMMON_DIR>\log\bulkloader`.

The logs include:

- message log—Records constraint violations specified in the *Error handling* section of the target table *Bulk Loader Options* tab.
- row log—Contains the data from the violating row. The data in the row log is delimited by the *Field delimiter* character specified on the *Bulk Loader Options* tab.

If you select *Clean up bulk loader directory after load*, Data Services deletes the data file and log files after loading completes. If you choose not to clean up the bulk loader directory or if your job results in errors captured in the logs, the software does not delete the data file and log files.

6.9.10 Bulk loading and reading in Teradata

SAP Data Services supports the following bulk loading and reading tools:

- Parallel Transporter
- FastLoad
- MultiLoad
- TPump
- Load Utility
- None (use ODBC)

> Note
>
> If your Job Server is on a UNIX platform, to take advantage of bulk loading on Teradata 13 databases you must set the required environment parameters in the file $LINK_DIR/bin/td_env.config. Instructions are documented inside the file.

For detailed information about Teradata options and their behavior in the Teradata environment, see the relevant Teradata product documentation.

Related Information

Reference Guide: Objects, Teradata source [page 952]

Reference Guide: Objects, Teradata target table options [page 989]

6.9.10.1 Bulk loader file options

For all bulk loader methods, you can use staging data files or named pipes. These *File option* types are on the *Bulk Loader Options* tab of the target table editor. This section describes how each file option works.

6.9.10.1.1 Data file

The Data file option loads a large volume of data by writing to a data file that it passes to the Teradata server. SAP Data Services runs bulk-loading jobs using a staging data file as follows:

1. It generates staging data file(s) containing data to be loaded into a Teradata table.
2. It generates a loading script to be used by Teradata Parallel Transporter. The script defines read and load operators.
3. If you use Teradata Parallel Transporter, the read operator reads the staging data file, then passes the data to the load operator, which loads data into the Teradata table.

6.9.10.1.2 Generic named pipe

The *Generic named pipe* file option loads a large volume of data by writing to a pipe from which Teradata reads. SAP Data Services runs bulk-loading jobs using a generic named pipe as follows:

1. It generates a script that Teradata Parallel Transporter uses to load the database.
2. It creates a pipe to contain the data to load into a Teradata table.
   On UNIX, the pipe is a FIFO (first in, first out) file that has a name of this format:

   ```
   /temp/<filename>.dat
   ```

   On Windows, the name has this format:

   ```
   \\.\pipe\<datastorename_ownername_tablename_loadernum>.dat
   ```
3. It executes the loading script. If you use Teradata Parallel Transporter, the script starts Teradata Parallel Transporter and defines read and load operators.
4. It writes data to the pipes.
5. Teradata Parallel Transporter connects to the pipes. Then the read operator reads the named pipe and passes the data to the load operator, which loads the data into the Teradata table.

6.9.10.1.3 Named pipes access module

The *Named pipes access module* file option loads a large volume of data by writing to a pipe from which Teradata reads. SAP Data Services runs bulk-loading jobs using a named pipe access module as follows:

1. Data Services generates a script that Teradata Parallel Transporter uses to load the database. The script starts Teradata Parallel Transporter and defines read and load operators.
2. Teradata (Parallel Transporter or non-Parallel Transporter utility) creates named pipes to contain the data to load into a Teradata table.
   On UNIX, the pipe is a FIFO (first in, first out) file that has name of this format:

   ```
   /temp/<filename>.dat
   ```

   On Windows, the name has this format:

   ```
   \\.\pipe\<datastorename_ownername_tablename_loadernum>.dat
   ```
3. Data Services connects to the pipes and writes data to them.

   > **Note**
   >
   > When Data Services tries to connect to the pipes, Teradata Parallel Transporter might not have yet created them. Data Services tries to connect every second for up to 30 seconds. You can increase the 30-second connection time to up to 100 seconds as follows: In the Designer, select ▶ *Tools* › *Options* › *Job Server* › *General* ◀ and enter the following:

   Section: **al_engine**
   Key: **NamedPipeWaitTime**
   Value: **<nn>**
   (**<nn>** is from 30 to 100)
4. The Teradata Parallel Transporter read operator reads the named pipe and passes the data to the load operator, which loads the data into the Teradata table.

6.9.10.2 When to use each Teradata bulk-loading method

SAP Data Services supports multiple bulk-loading methods for Teradata on Windows and UNIX. The following table lists the methods and file options that you can select, depending on your requirements.

Bulk loader method	File Option	Advantages	Restrictions
Parallel Transporter	Data file	• Can use Data Services parallel processing. • Data Services creates the loading script.	• The Teradata Server Tools and Utilities must be Version 7.0 or later. • If you use TTU 7.0 or 7.1, see the *Release Notes*.
	Generic named pipe	• Provides a fast way to bulk load because: ○ As soon as Data Services writes to a pipe, Teradata can read from the pipe. ○ Can use Data Services parallel processing. ○ On Windows, no I/O to an intermediate data file occurs because a pipe is in memory • Data Services creates the loading script.	• A job that uses a generic pipe is not restartable. • The Teradata Server Tools and Utilities must be Version 7.0 or later. • If you use TTU 7.0 or 7.1, see the *Release Notes*.
	Named pipe access module	• The job is restartable. • Provides a fast way to bulk load because: ○ As soon as Data Services writes to a pipe, Teradata can read from the pipe. ○ Can use Data Services parallel processing. ○ On Windows, no I/O to an intermediate data file occurs because a pipe is in memory. • Data Services creates the loading script.	• The Teradata Server Tools and Utilities must be Version 7.0 or later. • If you use TTU 7.0 or 7.1, see the *Release Notes*.

Bulk loader method	File Option	Advantages	Restrictions
Load utility	Data file	Load utilities are faster than INSERT statements through the ODBC driver.	● User must provide the loading script. ● Cannot use Data Services parallel processing
	Generic named pipe	● Load utilities are faster than INSERT statements through the ODBC driver. ● Named pipes are faster than data files because: ○ As soon as Data Services writes to a pipe, Teradata can read from the pipe. ○ On Windows, no I/O to an intermediate data file occurs because a pipe is in memory.	● User must provide the loading script. ● Cannot use Data Services parallel processing ● A job that uses a generic pipe is not restartable.
	Named pipes access module	● Load utilities are faster than INSERT statements through the ODBC driver. ● Named pipes should be faster than data files because: ○ As soon as Data Services writes to a pipe, Teradata can read from the pipe. ○ On Windows, no I/O to an intermediate data file occurs because a pipe is in memory ● The job is restartable.	● User must provide the loading script. ● Cannot use Data Services parallel processing.
FastLoad, MultiLoad, and TPump	Data file	Load utilities are faster than INSERT or UPSERT statements through the ODBC driver. Data Services creates the loading script.	Cannot use Data Services parallel processing
	Generic named pipe	● Load utilities are faster than INSERT or UPSERT statements through the ODBC driver.	● Cannot use Data Services parallel processing

Bulk loader method	File Option	Advantages	Restrictions
		• Named pipes are faster than data files because: ○ As soon as Data Services writes to a pipe, Teradata can read from the pipe. ○ On Windows, no I/O to an intermediate data file occurs because a pipe is in memory. • Data Services creates the loading script.	• A job that uses a generic pipe is not restartable.
	Named pipes access module	• Load utilities are faster than INSERT or UPSERT statements through the ODBC driver. • Named pipes should be faster than data files because: ○ As soon as Data Services writes to a pipe, Teradata can read from the pipe. ○ On Windows, no I/O to an intermediate data file occurs because a pipe is in memory. • The job is restartable. • Data Services creates the loading script.	Cannot use Data Services parallel processing.
None (use ODBC)	Uses Teradata ODBC driver to send separate SQL INSERT statements to load data.	INSERT statements through the ODBC driver are simpler to use than a data file or pipe.	This method does not bulk-load data.

Related Information

Designer Guide: Recovery Mechanisms, Automatically recovering jobs [page 721]

6.9.10.3 Parallel Transporter method

SAP Data Services supports Teradata's Parallel Transporter, an ETL tool that consolidates bulk-loading utilities into a single interface.

When you use the Parallel Transporter method, you can leverage Data Services' powerful parallel processing capabilities to specify a number of source and target options including the number of files or pipes for the software to use when processing large quantities of data.

6.9.10.3.1 Source performance tuning

Teradata source options are on the *Teradata options* tab in the source table editor. Here you can select the reading mode plus a number of advanced options.

You can tune the following options in the Teradata source editor to improve performance:

Option	Description
Maximum number of sessions	For large volumes of data, more sessions allows Data Services to read more data parallel. Ideally this number should equal the number of AMPs.
Number of export operator instances	When reading data in parallel, it can be consumed by multiple export instances for better performance. Ideally this value should equal the number of CPUs.
Parallel process threads	These internal threads break buffered data into rows and columns, which can improve performance by maximizing CPU usage on the Job Server computer. Ideally this number should equal the number of CPUs.

Related Information

Reference Guide: Teradata source [page 952]

6.9.10.3.1.1 Special considerations

Be aware of the following limitations when using the Teradata Parallel Transporter.

- It is not always possible to use the Parallel Transporter method to read from a source. In certain situations, Teradata does not support certain SQL constructs and you must use the ODBC method instead. In the following scenarios, Data Services will automatically switch the source mode to ODBC regardless of the actual mode selected on the *Teradata options* tab in the source editor:
  - The WHERE clause of a Query contains `<primary key>=<value>` predicate(s). Such a primary key can be a single column or a composite key.
  - The input schema contains columns of the LOB (CLOB or BLOB) data type.

- Unique secondary index columns are not allowed in the WHERE clause of a Query. However, because Data Services does not have the information as to whether a WHERE clause predicate is part of a unique secondary index, the WHERE clause gets pushed down to the Parallel Transporter reader. In this situation, a run-time error will occur, and you can manually set the source mode to *None* (ODBC).
- Database functions that are pushed down for ODBC readers are also pushed down for Parallel Transporter except for the functions Year and Month.
- Parallel Transporter does not accept parameterized SQL. As a result of this restriction, if a Teradata table is the inner loop of a join, the table will always be cached.
- Readers generated from Table Comparison transform and Lookup function families do not use Parallel Transporter.
- When multiple Teradata readers are optimized by Data Services by collapsing into one, Data Services uses Parallel Transporter whenever possible. When not possible, ODBC is used instead.

6.9.10.3.2 Target performance tuning

SAP Data Services provides the option for parallel processing when you bulk load data using the Parallel Transporter method.

In the target table editor using a combination of choices from the *Options* and *Bulk Loader Options* tabs, you can specify the number of data files or named pipes as well as the number of read and load Operator Instances. The *Number of Loaders* option distributes the workload while the Operators perform parallel processing.

In the target table *Options* tab, specify the *Number of Loaders* to control the number of data files or named pipes that Data Services or Parallel Transporter generates. Data Services writes data to these files in batches of 999 rows. For example, if you set *Number of Loaders* to 2, the software would generate two data files, writing 999 rows to the first file, then writing the next 999 rows to the second file. If there are more rows to process, the software continues, writing to the first file again, then the second, and so forth.

On the *Bulk Loader Options* tab, specify the number of instances in the loading scripts. If you set *Number of DataConnector instances* to 2 and *Number of instances* to 2, Parallel Transporter will assign the first read operator instance to read one data file and the other instance to read another data file in parallel. The DataConnector (read operator) instances then pass the data to the load operator instances for parallel loading into Teradata.

The Parallel Transporter uses a control file to read staging files or pipes and load data.

> **Note**
>
> Performance uaing this type of parallel loading depends on a number of factors such as distribution of incoming data and underlying DBMS capabilities. Under some circumstances, it is possible that specifying parallel loaders can be detrimental to performance. Always test the parallel loading process before moving to production.

6.9.10.3.2.1 Configuring the bulk loader for parallel processing

1. On the target table *Options* tab, specify the *Number of loaders* to control the number of data files or named pipes. Data Services will write data to these files in batches of 999 rows.

2. On the *Bulk Loader Options* tab, for *Bulk loader* choose *Parallel Transporter*.
3. For *File Option*, choose the type of file (*Data file*, *Generic named pipe*, or *Named pipes access module*) to contain the data to bulk load.
4. If you chose *Data file* or *Generic named pipe* in *File Option*, specify the number of read and load instances in the loading scripts.

   If you set *Number of instances* to 2 (load operators) and *Number of DataConnector instances* to 2 (read operators), Parallel Transporter will assign the first read operator instance to read one data file and the other instance to read another data file in parallel. The read operator instances then pass the data to the load operator instances for parallel loading into Teradata.

   > Note
   >
   > If you chose *Data file*, the value you specify for DataConnector instances (read operators) should be less than or equal to the number of data files.

5. If you chose *Named pipes access module* for *File Option*, specify *Number of instances* (load operators) in the loading scripts.

   Teradata uses the value you specify in *Number of loaders* to determine the number of read operator instances, as well as the number of named pipes. The DataConnector instances is not applicable when you use Named Pipes Access Module.

   For example, if you set *Number of loaders* to 2, Parallel Transporter generates two named pipes and assigns one read operator instance to read from one pipe and the other instance to read the other pipe in parallel. If you set *Number of instances* to 2 (load operators), the read operator instances pass the data to the load operator instances for parallel loading into Teradata.
6. If you specified *Named pipes access module* for *File Option*, you can override the default settings for the following Teradata Access Module parameters: Log directory, Log level, Block size, Fallback file name, Fallback directory, Signature checking.

   The Teradata Access Module creates a log file to record the load status and writes information to fallback data files. If the job fails, the Teradata Access Module uses the fallback data files to restart the load. The Access Module log file differs from the build log that you specify in the *Log directory* option in the Teradata datastore.

   > Note
   >
   > Data Services sets the bulk loader directory as the default value for both Log Directory and Fallback Directory.

   For more information about these parameters, see the relevant Teradata tools and utilities documentation.

Related Information

Reference Guide: Objects, Teradata target table options [page 989]

6.9.10.4 Teradata standalone utilities

In addition to the Parallel Transporter interface, SAP Data Services supports several Teradata utilities that load to and extract from the Teradata database. Each load utility is a separate executable designed to move data into a Teradata database. Choose from the following bulk loader utilities:

Utility	Description
FastLoad	Loads unpopulated tables only. Both the client and server environments support FastLoad. Provides a high-performance load (inserts only) to one empty table each session.
MultiLoad	Loads large quantities of data into populated tables. MultiLoad also supports bulk inserts, updates, upserts, and deletions against populated tables.
TPump	Uses standard SQL/DML to maintain data in tables. It also contains a method that you can use to specify the percentage of system resources necessary for operations on tables. Allows background maintenance for insert, update, upsert, and delete operations to take place at any time you specify. Used with small data volumes.
Load Utility	Invokes one of the above utilities (MultiLoad, FastLoad, or TPump) with the interface prior to Data Services version 11.5.1.

6.9.10.4.1 FastLoad

This procedure describes how to bulk load a table using the Teradata FastLoad utility.

1. Ensure that your Teradata datastore specifies a value in *TdpId* (Teradata Director Program Identifier). This option identifies the name of the Teradata database to load and is mandatory for bulk loading.
2. In the Bulk Loader Options tab of the target table editor, choose *FastLoad* in the *Bulk loader* drop-down list.
3. For *File option*, choose the type of file (*Data file*, *Generic named pipe*, or *Named pipes access module*) to contain the data to bulk load.
4. You can specify the following FastLoad parameters:

FastLoad parameter	Description
Data encryption	Encrypt data and requests in all sessions used by the job. The default is not to encrypt all sessions.
Print all requests	Prints every request sent to the Teradata database. The default is not to reduce print output.
Buffer size	Number of kilobytes for the output buffer that FastLoad uses for messages to the Teradata database. The default is 63 kilobytes which is also the maximum size.
Character set	Particular mapping between characters and byte strings (such as ASCII or UTF-8).

For more information about these parameters, see the Teradata FastLoad Reference.

5. In *Attributes*, you can usually use the default settings for the following attributes in the FastLoad script that SAP Data Services generates.

Script attribute	Description
AccountId	Identifier, of up to 30 characters, associated with the user name that will logon to the Teradata database.
CheckpointRate	The number of rows sent to the Teradata database between checkpoint operations. The default is not to checkpoint.
ErrorLimit	Maximum number of rejected records that Teradata can write to the error table 1 while inserting into a FastLoad table.
ErrorTable1	FastLoad uses this table to store records that were rejected for errors other than unique primary index or duplicate row violation.
ErrorTable2	FastLoad uses this table to store records that violated the unique primary index constraint.
MaxSessions	Maximum number of FastLoad sessions for the load job.
MinSessions	Minimum number of FastLoad sessions required for the load job to continue.
TenacityHours	Number of hours that the FastLoad utility continues trying to logon when the maximum number of load jobs are already running on the Teradata database.
TenacitySleep	Number of minutes that the FastLoad utility waits before it retries a logon operation. The default is six minutes.

> i Note
>
> By default, Data Services uses the bulk loader directory to store the script, data, error, log, and command (bat) files.

For more information about these parameters, see the Teradata FastLoad Reference.

6. If you specified *Data file* for *File Option*, you can increase the *Number of loaders* on the *Options* tab, which increases the number of data files. The software can use parallel processing to write data to multiple data files in batches of 999 rows.

   If you specified *Generic named pipe* or *Named pipes access module*, Data Services supports only one loader and disables the *Number of loaders* option.

Related Information

Reference Guide: Objects, Database datastores (Teradata) [page 868]

6.9.10.4.2 MultiLoad

This procedure describes how to bulk load a table using the Teradata MultiLoad utility.

1. Ensure that your Teradata datastore specifies a value in *TdpId* (Teradata Director Program Identifier). This option identifies the name of the Teradata database to load and is mandatory for bulk loading.
2. In the Bulk Loader Options tab of your target table editor, choose *MultiLoad* in the *Bulk loader* drop-down list.
3. In *File Option*, choose the type of file (*Data File*, *Generic named pipe*, or *Named pipes access module*) to contain the data to bulk load. The default is *Data File*.
4. You can specify the following MultiLoad parameters:

MultiLoad parameter	Short description
Reduced print output	The default is not to reduce print output.
Data Encryption	The default is not to encrypt all sessions.
Character set	Particular mapping between characters and byte strings (such as ASCII or UTF-8).

For more information about these parameters, see the Teradata MultiLoad Reference.

5. In *Attributes*, you can usually use the default settings for the following attributes in the MultiLoad script that SAP Data Services generates.

Script attribute	Short description
LogTable	Table in which Teradata stores the load job status. Specify the restart log table that will maintain the checkpoint information for your MultiLoad job.
AccountId	Identifier, of up to 30 characters, associated with the user name that will logon to the Teradata database.
WorkTable	Teradata uses this table to stage input data.
ErrorTable1	Teradata uses this table to store errors that it detects during the acquisition phase of the MultiLoad import task.
ErrorTable2	Teradata uses this table to store errors that it detects during the application phase of the MultiLoad import task.
ErrorLimit	Maximum number of rejected records that Teradata can write to the error table 1 during the acquisition phase of the MultiLoad import task. If used with ErrorPercentage, *ErrorLimit* specifies the number of records that must be sent to the Teradata database before *ErrorPercentage* takes effect.
ErrorPercentage	Approximate percentage (expressed as an integer) of total records sent so far (*ErrorLimit*) to the Teradata database that the acquisition phase might reject.
CheckpointRate	Interval between checkpoint operations during the acquisition phase. Express this value as either: ○ The number of rows read from your client system or sent to the Teradata database. ○ An amount of time in minutes.
MaxSessions	Maximum number of MultiLoad sessions for the load job.
MinSessions	Minimum number of MultiLoad sessions required for the load job to continue.
TenacityHours	Number of hours that the MultiLoad utility continues trying to logon when the maximum number of load jobs are already running on the Teradata database.

Script attribute	Short description
TenacitySleep	Number of minutes that the MultiLoad utility waits before it retries a logon operation. The default is six minutes.
TableWait	Number of hours that MultiLoad continues trying to start when one of the target tables is being loaded by some other job.
AmpCheck	Specifies how MultiLoad should respond when an Access Module Processor (AMP) is down.
IgnoreDuplicate	Select IgnoreDuplicate to not place duplicate rows in error table 2. The default is to load the duplicate rows.

Note

By default, Data Services uses the bulk loader directory to store the script, data, error, log, and command (bat) files.

For more information about these parameters, see the Teradata MultiLoad Reference.

6. If you specified *Data file* in *File Option*, you can increase the *Number of loaders* in the *Options* tab which increase the number of data files. Data Services can use parallel processing to write data to multiple data files in batches of 999 rows.

   If you specified *Generic named pipe* or *Named pipes access module*, Data Services supports only one loader and disables the *Number of loaders* option.

Related Information

Reference Guide: Objects, Database datastores (Teradata) [page 868]
Reference Guide: Objects, Target tables (Teradata target table options) [page 963]

6.9.10.4.3 TPump

This procedure describes how to bulk load a table using the Teradata TPump utility.

1. Ensure that your Teradata datastore specifies a value in *TdpId* (Teradata Director Program Identifier). This option identifies the name of the Teradata database to load and is mandatory for bulk loading.
2. On the *Bulk Loader Options* tab of the target table editor, choose *TPump* in the *Bulk loader* drop-down list.
3. For *File Option*, choose the type of file (*Data file*, *Generic named pipe*, or *Named pipes access module*) to contain the data to bulk load.
4. You can specify the following TPump parameters:

FastLoad parameter	Short description
Reduced print output	Reduce the print output of TPump to the minimal information required to determine the success of the job. The default is not to reduce print output.

FastLoad parameter	Short description
Retain Macros	Keep macros that were created during the job run. You can use these macros as predefined macros for subsequent runs of the same job.
Data Encryption	Encrypt data and requests in all sessions used by the job. The default is not to encrypt all sessions.
Number of buffers	Number of request buffers that TPump uses for SQL statements to maintain the Teradata database.
Character set	Particular mapping between characters and byte strings (such as ASCII or UTF-8).
Configuration file	Configuration file for the TPump job.
Periodicity value	Controls the rate at which TPump transfers SQL statements to the Teradata database. Value can be between 1 and 600, which specifies the number of periods per minute. The default value is 4 15-second periods per minute.
Print all requests	Turns on verbose mode which provides additional statistical data in addition to the regular statistics.

For more information about these parameters, see the Teradata Parallel Data Pump Reference.

5. In *Attributes*, you specify Data Services parameters that correspond to Teradata parameters in TPump commands. You can usually use the default settings for the following parameters in the TPump script that the software generates.

TPump command	Data Services parameter in Attributes pane	Description
NAME	AccountId	Identifier, of up to 30 characters, associated with the user name that will logon to the Teradata database.
BEGIN LOAD	Append	Use the error table specified in ErrorTable. If the table does not exist, TPump creates it. If the structure of the existing error table is not compatible with the error table TPump creates, the job will run into an error when TPump tries to insert or update the error table.
BEGIN LOAD	CheckpointRate	Number of minutes between checkpoint operations. Value must be an unsigned integer from 0 through 60, inclusive. The default is to checkpoint every 15 minutes.
BEGIN LOAD	ErrorLimit	Maximum number of rejected records that TPump can write to the error table while maintaining a table. The default is no limit. If you specify ErrorPercentage , *ErrorLimit* specifies the number of records that must be sent to the Teradata database before *ErrorPercentage* takes effect. For example, if ErrorLimit is 100 and ErrorPercentage is 5, 100 records must be sent to the Teradata database before the approximate 5% rejection limit is applied. If only 5 records were rejected when the 100th record is sent, the limit is not ex-

TPump command	Data Services parameter in Attributes pane	Description
		ceeded. However, if six records were rejected when the 100th record is sent, TPump stops processing because the limit is exceeded.
BEGIN LOAD	ErrorPercentage	Integer value that represents the approximate percent of the total number of records sent to the Teradata Database that might be rejected during the TPump task. You cannot specify this parameter without ErrorLimit.
BEGIN LOAD	ErrorTable	Name of the table in which TPump stores information about errors and the rejected records.
EXECUTE	ExecuteMacro	Name of macro to execute. Using predefined macros saves time because TPump does not need to create and drop new macros each time you run a TPump job script.
DML LABEL	Ignore duplicate inserts	Select *Ignore duplicate inserts* to not place duplicate rows in the error table.
NAME	JobName	Character string that identifies the name of a job. The maximum length is 16 characters.
BEGIN LOAD	Latency	Number of seconds that the oldest record resides in the buffer before TPump flushes it to the Teradata database. Value cannot be less than one second. If the *SerializeOn* is not specified, only the current buffer can possibly be stale. If you specify *SerializeOn*, the number of stale buffers can range from zero to the number of sessions.
Other TPump commands	LogTable	Name of the table to use to write checkpoint information that is required for the safe and automatic restart of a TPump job. The default name has the following format: `<owner>.<table>_LT`
BEGIN LOAD	MacroDatabase	Name of database to contain any macros TPump uses or builds. The default is to place macros in the same database that contains the TPump target table.
BEGIN LOAD	MaxSessions	Maximum number of sessions for TPump to use to update the Teradata database. SAP Data Services uses a default of 3.
BEGIN LOAD	MinSessions	Minimum number of sessions for TPump to use to update the Teradata database.
BEGIN LOAD	NoDrop	Do not drop the error table, even if it is empty, at the end of a job. You can use *NoDrop* with *Append* to persist the error table, or you can use it alone.

TPump command	Data Services parameter in Attributes pane	Description
BEGIN LOAD	NoMonitor	Prevents TPump from checking for statement rate changes from, or update status information for, the TPump Monitor.
IMPORT INFILE	NoStop	Prevents TPump from terminating because of an error associated with a variable-length record.
BEGIN LOAD	Pack	Number of SQL statements to pack into a multiple-statement request. The default is 20 statements per request. The maximum value is 600.
BEGIN LOAD	PackMaximum	Select PackMaximum to have TPump dynamically determine the number of records to pack within one request. The maximum value is 600.
BEGIN LOAD	Rate	Initial maximum rate at which TPump sends SQL statements to the Teradata database. Value must be a positive integer. If unspecified, Rate is unlimited.
BEGIN LOAD	Robust	Specifies whether or not to use robust restart logic. Value can be *YES* or *NO*. ○ *NO* specifies simple restart logic, which cause TPump to begin where the last checkpoint occurred in the job. TPump redoes any processing that occurred after the checkpoint. ○ *YES* specifies robust restart logic, which you would use for DML statements that change the results when you repeat the operation. Examples of such statements include the following: INSERTs into tables which allow duplicate rows `UPDATE foo SET A=A+1...`
BEGIN LOAD	SerializeOn	Specify a comma separated list of columns to use as the key for rows and guarantee that operations on these rows occur serially.
BEGIN LOAD	TenacityHours	Number of hours that the utility tries to log on sessions required to perform the TPump job. The default is four hours.
BEGIN LOAD	TenacitySleep	Number of minutes that TPump waits before it retries a logon operation. The default is six minutes.

Note

By default, SAP Data Services uses the bulk loader directory to store the script, data, error, log, and command (bat) files.

For more information about these parameters, see the Teradata Parallel Data Pump Reference.

6. If you specified *Data file* in *File Option*, you can increase the *Number of loaders* in the Options tab which increase the number of data files. The software can use parallel processing to write data to multiple data files in batches of 999 rows.

   If you specified Generic named pipe or Named pipe access module, Data Services supports only one loader and disables the Number of loaders option.

Related Information

Reference Guide: Objects, Database datastores (Teradata) [page 868]

6.9.10.4.4 Load Utility

To bulk load a Teradata table using the Load Utility:

1. On the *Bulk Loader Options* tab of your target table editor, choose *Load* in the *Bulk loader* drop-down list.
2. For *File Option*, choose the type of file (*Data File*, *Generic named pipe*, or *Named pipes access module*) to contain the data to bulk load.
3. Enter a command to be invoked by Data Services in the *Command line* text box. For example, `fastload<C:\tera_script\float.ctl`.
4. If you chose *Data file* for *File Option*, enter (or browse to) the directory path where you want the software to place your data file.
5. If you chose *Generic named pipe* or *Named pipes access module* for *File Option*, enter the pipe name.

6.9.10.5 Using the UPSERT bulk-loading operation

The purpose of the Teradata UPSERT operation is to update a row, but if no row matches the update, the row is inserted.

In SAP Data Services, you can only use the Teradata UPSERT operation with the following *Bulk loader* methods:

- MultiLoad
- TPump
- Parallel Transporter
  The UPSERT operation is available only with the following Operators:
  - Stream
  - Update

In Data Services, you enable UPSERT on the target table editor's *Bulk Loader Options* tab. In the *Data Services options* section, for *Bulk Operation*, select *Upsert* (the default is *Insert*).

The additional *Attributes* available when you select *Upsert* include:

- *Ignore missing updates*: Select whether or not to write the missing update rows into the error table. The default is *yes*.

- *Ignore duplicate updates*: Select whether or not to write an updated duplicate row into the error table. The default is *no*.

After selecting *Upsert*, note that you can also enable the *Use input keys* option on the target editor's *Options* tab.

Related Information

Reference Guide: Objects, Target tables [page 963]

6.9.11 Bulk loading using DataDirect's Wire Protocol SQL Server ODBC driver

Use the DataDirect's Wire Protocol SQL Server ODBC driver bulk load feature to quickly insert and update a large number of records into a database. You don't need to use a separate database load utility because the bulk load feature is built into the driver. DataDirect drivers are included in the Data Services installation.

For more detailed information about the Wire Protocol SQL Server ODBC driver, see the DataDirect documentation.

Some general considerations when using the bulk load feature:

- Enable the bulk load option only when you want to optimize load performance. Leaving the bulk load option enabled at all times could lead to undesired results or even corrupt data.
- Do not select the *Enable Bulk Load* option if any of the following Data Services loader options are enabled:
  - Include in Transaction
  - Use Overflow File
  - Auto Correct Load
  - Load Triggers
- You should create a different DSN and datastore when you are:
  - using the same SQL Server database server in different datastores.
  - using loaders that have different bulk load options.

6.9.11.1 Enabling the DataDirect bulk load feature in Windows

To enable the DataDirect bulk load feature for MS SQL Server on Windows, do the following:

1. Open the ODBC Data Source Administrator and go to the *Use DSN* tab. Click *Add*.
2. In the *Create New Data Source* window, select the DataDirect SQL Server Wire Protocol driver and click *Finish*. The *ODBC SQL Server Wire Protocol Driver Setup* window opens.
3. Enter driver setup information, such as the name of the data source, the host number, and so on.
4. On the Bulk tab, select the *Enable Bulk Loading* option.
5. Set the Bulk Options.

Option	Description
Keep Identity	Keeps source identity values.
Check Constraints	Checks constraints while data is being inserted into the database.
Keep Nulls	Keeps null values in the destination table.
Table Lock	Locks the table while the bulk copy operation is taking place. This option is checked by default.
Fire Triggers	Executes a trigger each time a row is being inserted into the database.
Bulk Binary Threshold (KB)	Maximum amount of data that you want exported to the bulk data file.
Batch Size	Number of rows you want the driver to send to the database at one time.
Bulk Character Threshhold (KB)	Maximum amount of character data that you want exported to the bulk data file.

6.9.11.2 Enabling the DataDirect bulk load feature in UNIX

To enable the DataDirect bulk load feature for MS SQL Server on UNIX, do the following:

1. Using a text editor, open the `odbc.ini` file associated with the DSN for which you are bulk loading.
2. Set the EnableBulkLoad option to 1.
3. Enter a BulkLoadOptions value. The value you enter depends on what options you enable.

Option	Description
Keep Identity	A value of 1 keeps source identity values.
Check Constraints	A value of 16 checks constraints while data is being inserted into the database.
Keep Nulls	A value of 64 keeps null values in the destination table.
Table Lock	A value of 2 locks the table while the bulk copy operation is taking place.
Fire Triggers	A value of 32 executes triggers when a row is being inserted into the database.

   Add the option values together and use the total as the BulkLoadOption value. For example, 16 (Check Constraints) + 32 (Fire Triggers) + 1 (Keep Identity) = 49.

   `BulkLoadOptions=49`.

4. Set the BulkLoadBatchSize option. This option sets the number of rows you want the driver to send to the database at one time. The default value is 1024.

Example

odbc.ini file

```
[ddsql]
Driver=/build/ds41/dataservices/DataDirect/odbc/lib/DAsqls25.so
Description=DataDirect 6.1 SQL Server Wire Protocol
AlternateServers=
AlwaysReportTriggerResults=0
AnsiNPW=1
ApplicationName=
ApplicationUsingThreads=1
AuthenticationMethod=1
```

```
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
BulkLoadOptions=2
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=ods
EnableBulkLoad=1
EnableQuotedIdentifiers=0
EncryptionMethod=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchTSWTZasTimestamp=0
FetchTWFSasTime=1
GSSClient=native
HostName=vantgvmwin470
HostNameInCertificate=
InitializationString=
Language=
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
MaxPoolSize=100
MinPoolSize=0
PacketSize=-1
Password=
Pooling=0
PortNumber=1433
QueryTimeout=0
ReportCodePageConversionErrors=0
SnapshotSerializable=0
TrustStore=
TrustStorePassword=
ValidateServerCertificate=1
WorkStationID=
XML Describe Type=-10
```

6.10 Other Tuning Techniques

The previous chapters describe the following tuning techniques:

- Maximizing push-down operations
- Using Caches
- Using Parallel Execution
- Distributing Data Flow Execution
- Using Bulk Loading

This section describes other tuning techniques that you can use to adjust performance:

- Source-based performance options
    - Join ordering
    - Minimizing extracted data
    - Using array fetch size

- Target-based performance options
  - Loading method
  - Rows per commit
- Job design performance options
  - Loading only changed data
  - Minimizing data type conversion
  - Minimizing locale conversion
  - Precision in operations

These techniques require that you monitor the change in performance carefully as you make small adjustments.

6.10.1 Source-based performance options

6.10.1.1 Join ordering

6.10.1.1.1 Join rank settings

You can use join rank to control the order in which sources are joined. Join rank indicates the rank of a source relative to other tables and files joined in a data flow. When considering join rank, the Data Services Optimizer joins sources with higher join ranks before joining sources with lower join ranks.

Join rank must be a non-negative integer. The default value is 0.

Although it is possible to set join rank in the Query editor FROM tab or in a source editor, the best practice is to specify the join rank directly in the Query editor.

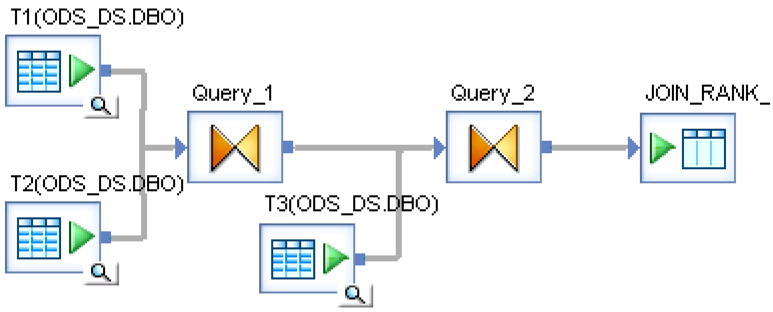
> i Note
>
> Join ranks set in sources are not displayed in the Query editor. If a join rank value is specified in the source, to find the value you must open the source editor.

The Data Services Optimizer gives preference to settings entered in the Query editor over settings entered in a source editor. If any one input schema has a join rank specified in the Query editor, then the Data Services Optimizer considers only Query editor join rank settings and ignores all source editor join rank settings.

Additionally, in a data flow containing multiple adjacent Query transforms, upstream join rank settings may be considered.

The join rank determination for multiple adjacent Query transforms can be complex because the Data Services Optimizer may combine these Query transforms into a single Query transform.

Consider a case where there is a join T1 inner join T2 in a query, Query_1; and the result of that join is used in another join in a downstream query, Query_2, as Query_1 inner join T3. The Optimizer would combine these two inner joins into a new Query, Query_2'.



Scenario 1

If the join rank values are set as follows:

Query editor	Table	Join rank
Query_1	T1	30
	T2	40
Query_2	Query_1 result set	10
	T3	20

The combined query, Query_2', would have the following join rank values:

Query editor	Table	Join rank
Query_2'	T1	30
	T2	40
	T3	41

The join rank value for T3 is adjusted to 41 because in the original Query_2 T3 has a higher join rank value than the result of T1 join T2 (Query_1 result set).

Scenario 2

In Query_2, if no join rank value is specified, then the default join rank of 0 is applied to both the Query_1 result set and T3. The join rank values are set as follows:

Query editor	Table	Join rank
Query_1	T1	30
	T2	40

Query editor	Table	Join rank
Query_2	Query_1 result set	not set (default=0)
	T3	not set (default=0)

The combined query, Query_2', would have the following join rank values:

Query editor	Table	Join rank
Query_2'	T1	30
	T2	40
	T3	40

The join rank value for T3 is adjusted to 40 because in the original Query_2 T3 has the same join rank value as the result of T1 join T2 (Query_1 result set).

Scenario 3

Assume join ranks are not set in the source tables. In Query_1, if no join rank value is specified, then the default join rank of 0 is applied to both T1 and T2. Values are set in the Query_2 Query editor as follows:

Query editor	Table	Join rank
Query_1	T1	not set (default=0)
	T2	not set (default=0)
Query_2	Query_1 result set	10
	T3	20

The combined query, Query_2', would have the following join rank values:

Query editor	Table	Join rank
Query_2'	T1	10
	T2	10
	T3	20

Scenario 4

If join rank values are not specified in the Query_1 and Query_2 Query editors, then the combined query, Query_2', would have no join rank values specified (default=0).

6.10.1.1.2 Join rank tips

For an inner join between two tables, in the Query editor assign a higher join rank value to the larger table and, if possible, cache the smaller table.

For a join between a table and file:

- If the file is small and can be cached, then assign it a lower join rank value and cache it.
- If you cannot cache the file, then assign it a higher join rank value so that it becomes the "outer table" in the join.

For a join between two files, assign a higher rank value to the larger file and, if possible, cache the smaller file.

6.10.1.1.3 About join ordering

The Data Services Optimizer determines the order it joins two or more tables based on the type of join and, where applicable, join rank. Although the join order has no effect on the actual result produced, controlling join order can often have a profound effect on the performance of producing the join result.

Join ordering is relevant only in cases where the Data Services engine performs the join. In cases where the query is pushed down to the database, the database determines how a join is done.

Join order in left outer joins

The result of a left outer join depends on the order of the sources. When considering left outer joins, the Data Services Optimizer does not change the join order from that specified in the Query editor. Since join order is not changed, join rank is not considered in the current query. However in a downstream query that uses the results of the left outer join as one of the sources, the join rank may be considered. Caching is implemented based on the cache settings specified in the Query editor.

Join order in inner joins

The result of an inner join is not dependent on the order of the sources. The Data Services Optimizer considers join rank and uses the source with the highest join rank as the left source. The Data Services Optimizer may choose to join tables in a different order than the order defined in the Query editor.

Viewing optimized join order

You can print a trace message to the Monitor log file which allows you to see the order in which the Data Services Optimizer performs the joins. This information may help you identify ways to improve performance.

To add the trace, select *Optimized Data Flow* from the list of traces in the *Trace* tab of the *Execution Properties* dialog.

View the results of the trace in the Trace log. The message begins `Join Order is:`.

Join order in real-time jobs

Using a left outer join is often preferable with real-time jobs because often you want the whole message passed on whether or not conditions a join looks for in the inner source exist.

In fact, if you do not use a left outer join to order joins for a message, the software will still process the message as if it has the highest join rank. The message source editor lacks a Join rank option because the software automatically gives it the highest join rank.

Related Information

Maximizing Push-Down Operations [page 2116]

6.10.1.1.4 Inner join example

When performing inner joins, the results of the join are identical no matter what order the joins are performed.

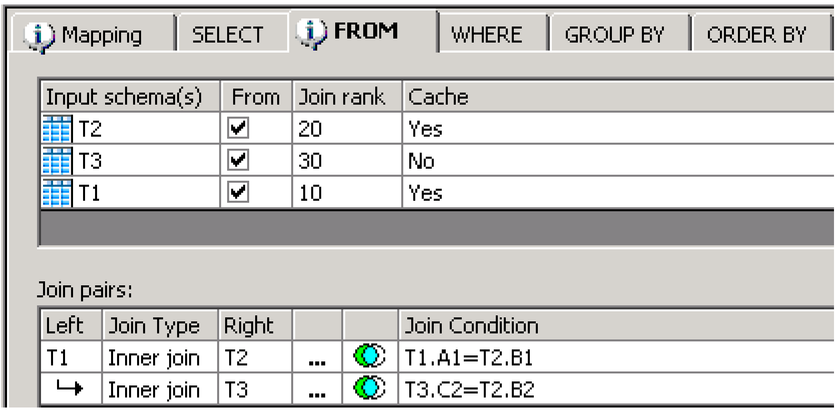
Because join rank is considered, for inner joins, the Data Services Optimizer may choose to join tables in a different order than the order defined in the Query editor.

When assigning join rank, also consider the effect of cache. If you assign a high join rank to a source, it will likely become the left (outer) source for the join. Generally, right (inner) tables are cached. When choosing to cache a source, ensure that you have sufficient memory or pageable cache.

Consider the following information:

Table	Column (data type)	Column (data type)	Number of Rows
T1	A1 (int)	A2 (int)	1,000,000
T2	B1 (int)	B2 (int)	1000
T3	C1 (int)	C2 (int)	1000

Suppose you wanted to create inner joins between T1, T2, and T3. Taking the number of rows in each table into consideration, in the Query editor you might specify join rank, cache, and join pairs as shown in the following screenshot:



Input schema(s)	From	Join rank	Cache
T2	☑	20	Yes
T3	☑	30	No
T1	☑	10	Yes

Join pairs:

Left	Join Type	Right			Join Condition
T1	Inner join	T2	...		T1.A1=T2.B1
↳	Inner join	T3	...		T3.C2=T2.B2

Although the join order is expressed as `T1 INNER JOIN T2 ON T1.A1=T2.B1 INNER JOIN T3 ON T3.C2=T2.B2`, the Data Services Optimizer determines the join order for inner joins. For the first join, the Optimizer takes source with the largest join rank, T3, as the left source and the table with the next highest join rank, T2, as the right source. The result of that inner join would then be joined with the remaining table, T1. The optimized SQL would appear as follows:

```
SELECT T1.A1, T3.C2
FROM (T3 INNER JOIN T2 ON (T3.C2=T2.B2))
INNER JOIN T1 ON (T1.A1=T2.B1)
```

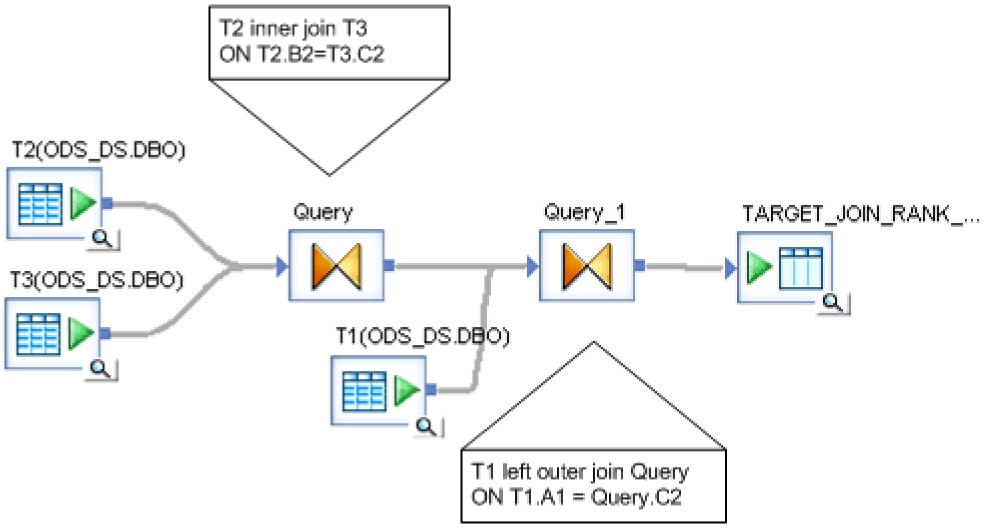
6.10.1.1.5 Join order with mixed joins example

Consider a case where you want to join T1 with the result of an inner join between T2 and T3.

```
FROM T1 LEFT OUTER JOIN (T2 INNER
          JOIN T3 ON T2.B2=T3.C2)
 ON T1.A1=T3.C2
```

Although you can specify mixed joins within the Query editor, you may specify the left table only for the first join. Subsequent joins take the result of the previous join as the left source. The data flow for this case must contain two queries. The first query will inner join T2 and T3; the second query is a left outer join with T1 as the left source and the result of the first query as the right source.

The data flow would look as follows:



In the Query_1 transform, C2 originates in T3.

Assume the source information in the following table:

Table	Column (data type)	Column (data type)	Number of Rows	Join Rank	Cache
T1	A1 (int)	A2 (int)	1,000,000	10	Yes
T2	B1 (int)	B2 (int)	1000	20	No
T3	C1 (int)	C2 (int)	1000	30	No

In the first query, T3 has the higher join rank so the Data Services Optimizer would perform the inner join as `T3 INNER JOIN T2 ON T3.C2 = T2.B2` even if the outer table specified in the Query editor is T2. In this case the cache setting for T2 is No, so the table would not be cached.

In the second query, T1 is left outer joined with the results of the first query. Join rank is not considered because this is a left outer join. In a scenario where join rank is relevant, the maximum join rank from the previous join is inherited. Additionally the result of the join T2 and T3 would be cached if either T2 or T3 has a cache setting of Yes.

6.10.1.2 Minimizing extracted data

The best way to minimize the amount of data extracted from the source systems is to retrieve only the data that has changed since the last time you performed the extraction. This technique is called changed-data capture.

Related Information

Designer Guide: Capturing Changed Data [page 738]

6.10.1.3 Using array fetch size

SAP Data Services provides an easy way to request and obtain multiple data rows from source databases. The array fetch size feature allows you to retrieve data using fewer requests, thereby significantly reducing traffic on your network. Tuning array fetch size can also reduce the amount of CPU use on the Job Server computer.

The array fetch feature lowers the number of database requests by "fetching" multiple rows (an array) of data with each request. Enter the number of rows to fetch per request in the *Array fetch size* option on any source table editor or SQL transform editor. The default setting is 1000, meaning that with each database request, The software will automatically fetch 1000 rows of data from your source database. The maximum array fetch size that you can specify is 5000 bytes.

It is recommended that you set the array fetch size based on network speed.

> i Note
>
> Higher array fetch settings will consume more processing memory proportionally to the length of the data in each row and the number of rows in each fetch.

Regardless of the array fetch setting, sources reading columns with an Oracle LONG data type cannot take advantage of this feature. If a selected data column is of type LONG, the array fetch size internally defaults to `1` row per request.

6.10.1.3.1 Setting the Array fetch size parameter

1. Use either a source table editor or an SQL transform editor.

   To use a source table editor:

   a) Double-click a source table in the Designer's workspace.

   b) In the *Performance* section of the *Source* tab, enter a number in the *Array fetch size* text box.

   To use an SQL transform editor:

   a) Double-click an SQL transform in the Designer's workspace.

   b) In the SQL transform editor, enter a number in the *Array fetch size* text box.

   Array Fetch Size indicates the number of rows returned in a single fetch call to a source table. The default value is 1000. This value reduces the number of round-trips to the database and can improve performance for table reads.

   The Array Fetch Size option does not support long column data types. If the *SELECT* list contains a long column, the software sets the *Array Fetch Size* to 1 and reads one row of data at a time from the database.
2. Click *OK*.

6.10.1.3.2 Tip

The optimal number for *Array fetch size* depends on the size of your table rows (the number and type of columns involved) as well as the network round-trip time involved in the database requests and responses. If your computing environment is very powerful, (meaning that the computers running the Job Server, related databases, and connections are extremely fast), then try higher values for *Array fetch size* and test the performance of your jobs to find the best setting.

6.10.2 Target-based performance options

6.10.2.1 Loading method

You can choose to use regular loading or bulk loading. For a regular load, the *Parameterized SQL* option is automatically selected when generating, parsing, and compiling the statement. By using parameterized SQL, the software can minimize these efforts by using one handle for a set of values instead of one handle per value.

Many databases do not support bulk loading with the following options; see the specific options for your target database in the *Reference Guide*.

- Auto-correct load
- Enable Partitioning
- Number of Loaders
- Full push down to a database
  The software automatically selects this optimizer process when the following conditions are met:
  - The source and target in a data flow are on the same database.
  - The database supports the operations in the data flow.

  If the optimizer pushes down source or target operations, then it ignores the performance options set for sources (Array fetch size, Caching, and Join rank) because it is not solely processing the data flow.
- Overflow file
- Transactional loading

To improve performance for a regular load (parameterized SQL), you can select the following options from the target table editor. Note that if you use one, you cannot use the others for the same target.

- Enable Partitioning
  Parallel loading option. The number of parallel loads is determined by the number of partitions in the target table.
- Number of Loaders
  Parallel loading option. The number of parallel loads is determined by the number you enter for this option.

Related Information

Push-down operations [page 2116]
Table partitioning [page 2138]
Bulk Loading and Reading [page 2164]
Reference Guide: Objects, Target tables [page 963]

6.10.2.2 Rows per commit

Rows per commit for regular loading defaults to 1000 rows. Setting the *Rows per commit* value significantly affects job performance. Adjust the rows per commit value in the target table editor's *Options* tab, noting the following rules:

- Do not use negative number signs and other non-numeric characters.
- If you enter nothing or `0`, the text box will automatically display `1000`.
- If you enter a number larger than `5000`, the text box automatically displays `5000`.

It is recommended that you set rows per commit between `500` and `2000` for best performance. You might also want to calculate a value. To do this, use the following formula:

`max_IO_size/row size` (in bytes)

For most platforms, `max_IO_size` is 64K. For Solaris, `max_IO_size` is 1024K.

Note that even with a value greater than one set for *Rows per commit*, SAP Data Services will submit data one row at a time if the following conditions exist:

- You are loading into a database (this scenario does not apply to Oracle databases), and have a column with a LONG datatype attribute.
- You are using an overflow file where the transaction failed. However, once all the rows are loaded successfully, the commit size reverts to the number you entered. In this case, depending on how often a load error happens, performance might be come worse than setting *Rows per commit* to `1`.

Related Information

Caching sources [page 2130]

6.10.3 Job design performance options

6.10.3.1 Loading only changed data

Identifying and loading only changed data is called changed-data capture (CDC), which includes only incremental data that has changed since the last refresh cycle. Performance improves because with less data to extract, transform, and load, the job typically takes less time.

Related Information

Designer Guide: Capturing Changed Data [page 738]

6.10.3.2 Minimizing data type conversion

SAP Data Services offers very robust and easy-to-use data type conversions via column mappings of different data types. It is recommended that you:

- Avoid unnecessary data conversions.
- Verify that Data Services is performing the implicit conversions (selected when you drag and drop columns from input to output schemas in the query transform) as expected. This can be done by looking at the warnings generated during job validation.

6.10.3.3 Minimizing locale conversion

If your jobs do not require the use of different or multi-byte locales, you can increase performance by ensuring that locales are single-byte and not mismatched.

6.10.3.4 Precision in operations

SAP Data Services supports the following precision ranges: 0-28, 29-38, 39-67, 68-96. Note that as you decrease precision, performance increases for arithmetic operations and comparison operations. In addition, when processing an arithmetic or boolean operation that includes decimals in different precision ranges, the software converts all to the highest precision range value because it cannot process more than one decimal data type precision range for a single operation. For example, if the software must perform an arithmetic operation for decimals with precision 28 and 38, it converts both to precision 38 before completing the operation.

7 Integrator Guide

7.1 Welcome to SAP Data Services

7.1.1 Welcome

SAP Data Services delivers a single enterprise-class solution for data integration, data quality, data profiling, and text data processing that allows you to integrate, transform, improve, and deliver trusted data to critical business processes. It provides one development UI, metadata repository, data connectivity layer, run-time environment, and management console—enabling IT organizations to lower total cost of ownership and accelerate time to value. With SAP Data Services, IT organizations can maximize operational efficiency with a single solution to improve data quality and gain access to heterogeneous sources and applications.

7.1.2 Documentation set for SAP Data Services

You should become familiar with all the pieces of documentation that relate to your SAP Data Services product. The latest Data Services documentation can be found on the *SAP Help Portal*.

Document	What this document provides
Adapter SDK Guide	Information about installing, configuring, and running the Data Services Adapter SDK
Administrator Guide	Information about administrative tasks such as monitoring, lifecycle management, security, and so on.
Customer Issues Fixed	Information about customer issues fixed in this release. ℹ Note In some releases, this information is displayed the Release Notes.
Designer Guide	Information about how to use Data Services Designer.
Documentation Map	Information about available Data Services books, languages, and locations.
Installation Guide for Windows	Information about and procedures for installing Data Services in a Windows environment.
Installation Guide for UNIX	Information about and procedures for installing Data Services in a UNIX environment.
Integrator Guide	Information for third-party developers to access Data Services functionality using web services and APIs.
Master Guide	Information about the application, its components and scenarios for planning and designing your system landscape. Information about SAP Information Steward is also provided in this guide.

Document	What this document provides
Management Console Guide	Information about how to use Data Services Administrator and Data Services Metadata Reports.
Performance Optimization Guide	Information about how to improve the performance of Data Services.
Reference Guide	Detailed reference material for Data Services Designer.
Release Notes	Important information you need before installing and deploying this version of Data Services.
Technical Manuals	A compiled, searchable, “master” PDF of core Data Services books: ● *Administrator Guide* ● *Designer Guide* ● *Reference Guide* ● *Management Console Guide* ● *Performance Optimization Guide* ● *Integrator Guide* ● *Supplement for J.D. Edwards* ● *Supplement for Oracle Applications* ● *Supplement for PeopleSoft* ● *Supplement for Salesforce.com* ● *Supplement for Siebel* ● *Supplement for SAP* ● *Workbench Guide*
Text Data Processing Extraction Customization Guide	Information about building dictionaries and extraction rules to create your own extraction patterns to use with Text Data Processing transforms.
Text Data Processing Language Reference Guide	Information about the linguistic analysis and extraction processing features that the Text Data Processing component provides, as well as a reference section for each language supported.
Tutorial	A step-by-step introduction to using Data Services.
Upgrade Guide	Information to help you upgrade from previous releases of Data Services and release-specific product behavior changes from earlier versions of Data Services to the latest release.
What's New	Highlights of new key features in this SAP Data Services release. This document is not updated for support package or patch releases.
Workbench Guide	Provides users with information about how to use the Workbench to migrate data and database schema information between different database systems.

In addition, you may need to refer to several Supplemental Guides.

Document	What this document provides
Supplement for SAP	Information about interfaces between Data Services, SAP Applications, SAP Master Data Services, SAP NetWeaver BW, and SAP Master Data Services.

Document	What this document provides
Supplement for SuccessFactors	Information about interfaces between Data Services and SuccessFactors.
Supplement for Salesforce.com	Information about how to install, configure, and use the SAP Data Services Salesforce.com Adapter Interface.
Supplement for J.D. Edwards	Information about interfaces between Data Services and J.D. Edwards World and J.D. Edwards OneWorld.
Supplement for Oracle Applications	Information about the interface between Data Services and Oracle Applications.
Supplement for PeopleSoft	Information about interfaces between Data Servicesand PeopleSoft.
Supplement for Siebel	Information about the interface between Data Services and Siebel.

We also include these manuals for information about SAP BusinessObjects Information platform services.

Document	What this document provides
Information platform services Administrator Guide	Information for administrators who are responsible for configuring, managing, and maintaining an Information platform services installation.
Information platform services Installation Guide for UNIX	Installation procedures for SAP BusinessObjects Information platform services on a UNIX environment.
Information platform services Installation Guide for Windows	Installation procedures for SAP BusinessObjects Information platform services on a Windows environment.

7.1.3 Accessing documentation

You can access the complete documentation set for SAP Data Services in several places.

7.1.3.1 Accessing documentation on Windows

After you install SAP Data Services, you can access the documentation from the Start menu.

1. Choose *Start* > *Programs* > *SAP Data Services 4.2* > *Data Services Documentation* > *All Guides*.
2. Click the appropriate shortcut for the document that you want to view.

7.1.3.2 Accessing documentation on UNIX

After you install SAP Data Services, you can access the documentation by going to the directory where the printable PDF files were installed.

1. Go to `<LINK_DIR>/doc/book/en/`.

2. Using Adobe Reader, open the PDF file of the document that you want to view.

7.1.3.3 Accessing documentation from the Web

You can access the complete documentation set for SAP Data Services from the SAP Business Users Support site.

To do this, go to *http://help.sap.com/bods*.

You can view the PDFs online or save them to your computer.

7.1.4 SAP information resources

A global network of SAP technology experts provides customer support, education, and consulting to ensure maximum information management benefit to your business.

Useful addresses at a glance:

Address	Content
Customer Support, Consulting, and Education services *http://service.sap.com/*	Information about SAP support programs, as well as links to technical articles, downloads, and online forums. Consulting services can provide you with information about how SAP can help maximize your information management investment. Education services can provide information about training options and modules. From traditional classroom learning to targeted e-learning seminars, SAP can offer a training package to suit your learning needs and preferred learning style.
Product documentation *http://help.sap.com/bods/*	SAP product documentation.
Supported Platforms (Product Availability Matrix) *https://service.sap.com/PAM*	Get information about supported platforms for SAP Data Services. Use the search function to search for Data Services. Click the link for the version of Data Services you are searching for.
SAP Data Services Community Network *http://scn.sap.com/community/data-services*	Get online and timely information about SAP Data Services, including forums, tips and tricks, additional downloads, samples, and much more. All content is to and from the community, so feel free to join in and contact us if you have a submission.
Blueprints *http://scn.sap.com/docs/DOC-8820*	Blueprints for you to download and modify to fit your needs. Each blueprint contains the necessary SAP Data Services project, jobs, data flows, file formats, sample data, template tables, and custom functions to run the data flows in your environment with only a few modifications.

Address	Content
SAPTerm *https://portal.wdf.sap.corp/go/sapterm*	SAP’s terminology database, the central repository for defining and standardizing the use of specialist terms.

7.2 Web service support

This section discusses both how an administrator can configure SAP Data Services through the Administrator to publish jobs as callable web services, and how an application developer can access those web services.

The software publishes web services from the Management Console Administrator. To use SAP Data Services as a web service, select the Web Services node in the Administrator's navigation tree. For general information on using the Administrator, see the *Management Console Guide*.

7.2.1 Overview

Web services are modular business applications based on open standards (WSDL, REST, SOAP, and XML Schema primarily) that allow integration among different applications and environments through the Internet. Web services allow parts of existing applications to be used by other applications.

For business intelligence (BI), you can use web services to accomplish the following:

- Access legacy systems
- Conduct computer-to-computer interaction over an internal or external web
- Allow applications constructed in different languages on different platforms to communicate with each other in an enterprise environment

SAP Data Services can:

- Publish any job as a callable web service (server functionality)
- Call published web services from within its jobs using the built-in web services datastore (client functionality)

If you have an application that also supports web services, you can use that application to run batch and real-time jobs or to publish your application's functionality to be called by data flows.

After you install the software, you can immediately start working with its client functionality because the built-in web services datastore is a web services client that provides access to a web services server from a data flow.

Related Information

Using SAP Data Services as a web service provider [page 2218]

Consuming external web services in SAP Data Services [page 2257]

7.2.2 Web services technologies

SAP Data Services web services are fully compliant with Web Services Interoperability (WS-I) Basic Profile 1.0, and support three Java Web Services technologies.

Web service technology	Description
SOAP	Connection protocol (envelope for XML messages).
WSDL	Language used to request a service and return replies (subset of XML).
XML Schema	Format used for the WSDL file.

The software supports SOAP 1.1, WSDL version 1.1, 2.0 and 2.1, and Apache Axis 1.1 (an industry-standard SOAP message handler and WSDL parser).

SAP Data Services is also compliant with the Microsoft .NET environment for web services. You can import the WSDL that the software generates into Visual Studio .NET and the web services datastore can call the WSDL that Visual Studio .Net generates.

SAP Data Services also supports the following for REST web services:

Web service technology	Description
REST	Design pattern based on HTTP connection protocol.
WADL	Language used to describe the request and response for a web provider.
XML Schema	Schema used to describe the input and output for a web service operation.
JSON/XML	Message format used by web services to communicate with a client.

ℹ Note

Currently, SAP Data Services only supports consuming external REST web service providers (for example, outbound web calls allowing you to call a provider). The software does not expose REST web service operations (for example, inbound web calls allowing you to access SAP Data Services).

Related Information

SOAP [page 2214]
REST web services [page 2216]

7.2.2.1 SOAP

SAP Data Services allows you to invoke real-time services using the following:

- Message Client API (which supports C++ and Java connections)
- TCP/IP
- proprietary XML using HTTP

In addition, the software supports the Simple Object Access Protocol (SOAP). SOAP is the industry standard from the World Wide Web Consortium (WC3.org) used to invoke network resources using XML over HTTP, HTTPS, and other standard protocols.

A SOAP gateway is built in to the Administrator. The software supports SOAP over HTTP and HTTPS protocols.

7.2.2.1.1 WSDL

Web Services Description Language (WSDL) is a subset of XML. It is used as a transport mechanism for XML messages. SAP Data Services publishes its jobs in WSDL based on configuration settings applied in the Administrator, and then developers can create a web services client based on the software's WSDL.

The software also publishes all comments entered into the Designer's *Job Descriptions* box with each job that is added to the WSDL file.

The WSDL file generated by the software includes tags (such as services, bindings, ports, and operations) that support the use of the SOAP protocol. Each tag uses a name that the software provides. For example:

- You select which jobs to publish in the web service named `DataServices_server`. In WSDL, a service is a set of business operations with connection endpoints.
- Binding names include `Connection_Operations`, `Batch_Jobs`, `Real-time_Services`, and `Batch_Job_Admin`. WSDL uses bindings to associate operations with ports.
- Operation names have a one-to-one relationship with the names of batch jobs or real-time services.

7.2.2.1.2 XML Schema

WSDL uses XML Schemas to define input and output message formats.

- For server functionality, if a real-time service was defined with DTDs, you will need to translate the DTD format into the XML Schema format.
- For client functionality, the web services datastore imports metadata into SAP Data Services using the XML Schema format only.

XML Schema formats are defined in the *types* element of the WSDL file.

> i Note
>
> When you import an XML schema for a real-time web service job, you should use a unique target namespace for the schema. When Data Services generates the WSDL file for a real-time job with a source or target schema that has no target namespace, it adds an automatically generated target namespace to the types section of the XML schema. This can reduce performance because Data Services must suppress the namespace information from the web service request during processing, and then reattach the proper namespace information before returning the response to the client.

7.2.2.1.3 UDDI

UDDI is a method of publishing comments and other reference information about jobs to an external web site. SAP Data Services does not publish information to a UDDI web site because most web service users work behind enterprise firewalls.

7.2.2.2 REST web services

Representational State Transfer (REST or RESTful) web service is a design pattern for the World Wide Web. Data Services allows you to call the REST server and then browse through and use the data the server returns. Unlike SOAP, which exposes functions, REST exposes resources (data). The REST web services supports JSON and XML data types.

Each function in REST web services uses a unique endpoint uniform resource identifier (URI). Data Services strictly requires each function to have a fixed schema defined by XSD (XML Schema Definition). Based on HTTP verbs, Data Services is then able to get or modify resources. Some of the more common HTTP verbs include GET (retrieves data), PUT (updates data), POST (creates new data), and DELETE (deletes data).

> **Note**
>
> Data Services does not support the HEAD HTTP verb. It does not expose header outputs in the Query transform.

In order to call the REST web service through Data Services, you need to provide a Web Application Description Language (WADL) file. This XML file describes methods and input and output schemas and provides Data Services with all the required parameters it needs to call REST functions.

Data Services allows you to do the following:

- Create a Web Services REST datastore. The datastore exposes functions that Data Services uses to call the server through HTTP communication.
- Parse the WADL file to expose REST functions.
- Browse a Web Services REST datastore to find and import functions.
- Use REST functions in a Query transform.
- Use JSON or XML as input and output for REST functions.
- Consume REST web services (outbound only) from the engine.

For more information, see "Web service datastores" in the *Designer Guide*.

Authorization schemas

Data Services supports the following authorization schemas. You must have knowledge of the provider to which you are connecting in order to obtain some configuration information, such as the consumer key.

- Basic
- Basic + CSRF (Cross-Site Request Forgery protection)

- Authorization Header (Custom Token, API key, and so on)
- OAuth 1.0 (2-legged)
- OAuth 2.0 (2-legged)

> **Note**
>
> For OAuth 2.0 based authentication, Data Services only supports client_credentials and password-based grant types. Data Services does not support the 3-legged model.

Related Information

WADL [page 2217]

7.2.2.2.1 WADL

You must provide Data Services with a well-organized WADL file. This file should follow the schema provided by the World Wide Web Consortium (WC3.org) at *http://www.w3.org/Submission/wadl/wadl.xsd* . There are various third party tools that you can use to create and edit this file.

The file should contain a Grammars element, which provides the location for input and output schemas. Data Services can then parse elements and store them in an internal repository for browsing and exposing operations.

A WADL file contains many components. The following table explains some of the more common elements of the schema:

Element	Description
Application	Forms the root of the WADL file and contains all the resources.
Doc	Describes an element. If provided in the param element, this information will be displayed as the function description in the Designer user interface.
Grammars	Provides the external location for input and output schemas.
Resources	Contains a list of resources.
Resource	Defines a unique URI.
Method	Defines the HTTP verb for selected resource.
Representation	Indicates the content type and respective element defined in grammar.
Param	Contains parameters that form a part of the URI or header.

Related Information

Designer Guide: To define a web services datastore [page 237]

7.2.2.2.2 HTTP Status Codes

HTTP Status Codes are standard response codes that a server returns to help you identify the cause of a problem.

An HTTP Status Code of 200 OK is the correct response (for example, the XML/JSON document matches the schema output schema) and is considered a success. An HTTP Status Code greater than or equal to 400 is considered an error and Data Services populates the AL_ERROR_MSG column with a server error message.

- AL_ERROR_NUM — returns error codes: 0 for success and a non-zero integer for failures
- AL_ERROR_MSG — returns an error message if AL_ERROR_NUM is not 0. Returns NULL if AL_ERROR_NUM is 0

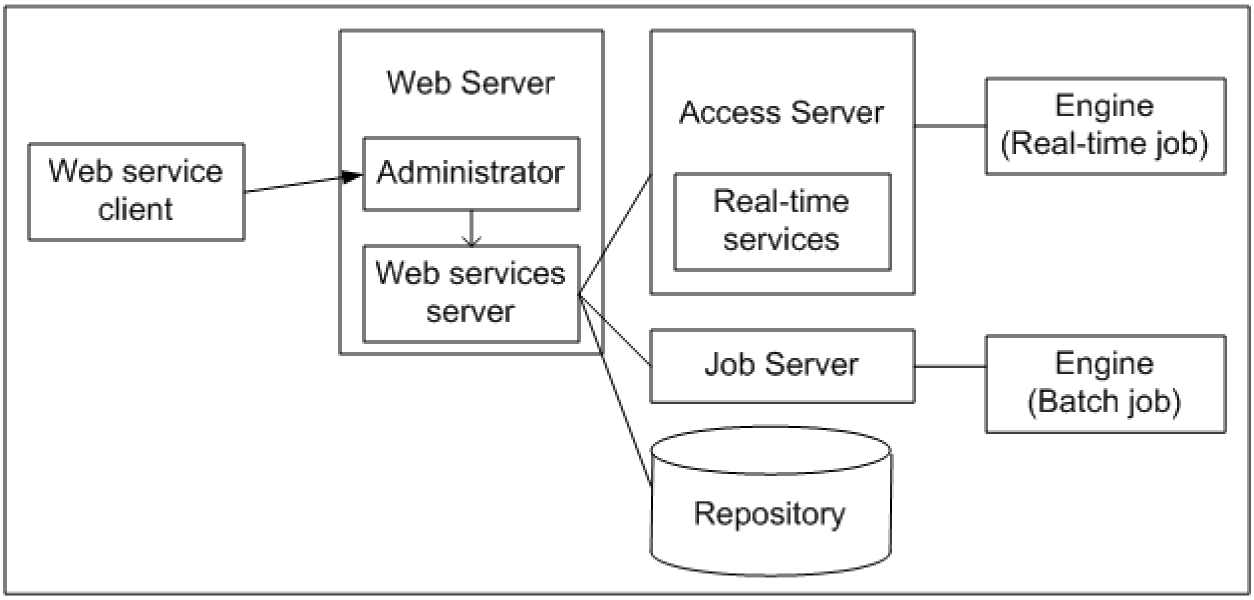
Table 183: Error codes for AL_ERROR_NUM column

Error code	Description
0	The call was successful and the server returned a 200 OK status.
1	NRDM to XML/JSON conversion failed. The engine was unable to successfully create the message payload. This can be caused by an invalid character in the input, and so on.
2	XML/JSON to NRDM conversion failed. Engine was unable to successfully convert the response payload. Possible causes include: • invalid XML/JSON • server did not return a document that matches the schema (the engine requires the XML/JSON document to exactly match the schema) • invalid characters (for example, control characters)
3	The call to the server failed. Possible causes include: • invalid URL • malformed URL • server does not exist • SSL certificate is missing • proxy or firewall does not allow Data Services to communicate with the URL • no internet communication • resource can't be found • lack of privileges to access the resource
4	An OAuth communication error. This error can be caused by all the items listed for error code 3, but are related to only OAuth and OAuth errors.

7.3 Using SAP Data Services as a web service provider

After the Administrator publishes batch or real-time jobs as web services, the web application hosts those web services. When an external application calls into SAP Data Services through web services, the application acts as a web services client accessing a web services server.

Web service clients call the published web services, pass in the appropriate parameters, and receive the results. The software routes calls to the appropriate Job Server and job for processing.



Web services might be used in the following example scenarios:

- Dynamically update an internal web site
  Suppose you have an internal web site that manages foreign exchange rate status worldwide for the Finance department. When foreign exchange rates change more than a certain percentage, a batch job updates exchange rates in your financial data mart. The rate change initiates a call to a web service that starts the appropriate batch ETL job.
- Solve a processing issue
  Suppose you have an existing Enterprise Application Integration (EAI) bus infrastructure and want to manage batch processes and EAI transactional processes from within the same infrastructure. The transactional processes are complex. Their staging is laid out in the order process. However, EAI work flows do not have the ability to run batch processes.
  The software can publish Web services that allow you to leverage EAI process management category tools (for example, webMethods Business Process Manager) to control and stage batch processes alongside its transactional processes.
  The work flows might call the software to:
  - Perform an initial load of a data mart for real-time reporting
  - Refresh the data cache depending on specified business criteria
  - Perform complex transforms on hierarchical objects for mapping data between ERP systems

7.3.1 WSDL basics

WSDL is a subset of XML that you can use to describe network services as a collection of endpoints capable of exchanging messages.

This table shows the elements in a WSDL file, and describes how those elements are used in the SAP Data Services-generated WSDL file.

Element Name	Description
definition	Root element
service	Used to group a set of related ports or endpoints to which a client application will connect. The software publishes a single service in the WSDL file it generates.
port	Defines a specific web service endpoint that a client can access. Each port has a unique name and a specific address used for binding. The software defines ports for web services as: ● `Connection_Operations`: used for authentication and ping ● `Real-time_Services`: used for real-time jobs exposed as web services ● `Batch_Jobs`: used for batch jobs exposed as web services (each batch jobs has its own operation) ● `Batch_Job_Admin`: used for administrative operations for batch jobs
portType	Defines a set of operations that a web service publishes. A portType is bound to a particular port. The binding specifies the protocol and data formation for the operations defined by a portType.
operation	Defines a specific function call. The software publishes each batch job and real-time service as an operation. It also publishes connection operations.
message	Defines the data to transmit. There is an input (request) message, which the web service receives from the client, and there is an output (response) message, which the web service sends back to the client.
type	Defines the data types used in messages sent to/from a web service.

Related Information

SoapAction element [page 2247]

7.3.1.1 Building a WSDL file

Use the information in the WSDL file produced by SAP Data Services to create an application that can access batch jobs and real-time services. Access the WSDL file by making web service client calls to it using a reference URL.

To view the WSDL file so that you can create your application, use the *Web Services* node of the Management Console Administrator, or open a browser window and search for:

http://**`<hostname:port>`**/DataServices/servlet/webservices?ver=2.0&wsdlxml

7.3.1.1.1 To configure web service information using the Administrator

1. Open the Administrator.
2. Log in with Administrator-level privileges. Users with Monitor-level privileges cannot configure web services.

   > i Note
   >
   > If you enable security for the WSDL file, SAP Data Services requires that web services clients use the user name and password of any user with Administrator-level privileges to access all published web services.

3. Add connections from Access Servers and repositories to view jobs in the Administrator.
4. If you plan to publish real-time jobs as web services, configure real-time jobs as real-time services.

   The software publishes the following as web services:

   - Real-time services enabled as web service operations in the Administrator
   - Batch jobs enabled as web service operations in the Administrator
   - Connection Operations
     - Ping - Used to ping Web services
     - Logon and Logout - Security operations that provide controlled access to Web service operations (if enabled).
5. In the Administrator's navigation tree, select *Web Services*.

   The *Web Services Status* page opens. This page lists Web service operations that are published in the WSDL. By default, only the Ping operation is automatically published.
6. Click the *Web Services Configuration* tab.

   Use the *Configuration* tab to open the *Web Services Configuration* page. Use this page to select jobs and real-time services to be published, enable/disable security for the WSDL file, and to enable/disable access to full batch job attributes.
7. From the pull-down menu, use *Add Real-time Service* or *Add Batch Job* to add jobs or services to the WSDL, and click *Apply*.

   On the *Add Real-time Service* page, real-time services are grouped by the Access Server for which the service is configured. To add a real-time service to the WSDL, select an Access Server or select *All*, select the check box in front of a real-time service name, and click *Add*.

   On the *Add Batch Job* page, jobs are grouped by the repository on which the job is stored. To add a job to the WSDL, select a repository or select *All*, select the check box in front of a job name, and click *Add*.
8. (Optional) On the *Web Services Configuration* page, select *Enable Session Security* and click *Apply* to enable security for the WSDL.

   Security for published operations is disabled by default.

   With security enabled, instead of making a single call to the Administrator to start a batch job or trigger a real-time service from an external application, clients must make at least three calls:

   - The first call logs in to the Administrator and gets a session ID.
   - The second call accesses a job or service using the session ID as an input parameter. Create a call for each job or service you want to access.
   - The final call logs out of the session.

9. (Optional) On the *Web Services Configuration* page, from the drop-down menu, select *Enable Job Attributes* to allow the input message for all the batch jobs you publish to include all options supported for submitting batch jobs from the Administrator. The following table lists elements added to the message:

Element	Description
job_system_profile	System profile used to run the job.
sampling_rate	Monitor sample rate (# of seconds).
auditing	Enable auditing (true or false).
recovery	Enable recovery (true or false).
job_server	Job Server or Server Group.
trace	Trace option to be enabled. You must specify an option to enable tracing. This element can be repeated for as many trace options as you require. The WSDL defines values for the trace option and includes the following (all options on the batch job submit page of the administrator): ○ job_trace_all ○ job_trace_row ○ job_trace_session ○ job_trace_workflow ○ job_trace_dataflow ○ job_trace_transform ○ job_trace_usertransform ○ job_trace_userfunction ○ job_trace_abapquery ○ job_trace_sqlfunctions ○ job_trace_sqlreaders ○ job_trace_sqlloaders ○ job_trace_optimized_dataflow ○ job_trace_table ○ job_trace_script ○ job_trace_ascomm ○ job_trace_rfc_function ○ job_trace_table_reader ○ job_trace_idoc_file ○ job_trace_adapter ○ job_trace_communication ○ job_trace_parallel_execution ○ job_trace_audit
distribution_level	You can distribute the execution of a job or a part of a job across multiple Job Servers within a Server Group to better balance resource-intensive operations.

Element	Description
	You can specify the following values on the distribution level option when you execute a job: ○ Job: A job can execute on an available Job Server. ○ Data flow: Each data flow within a job can execute on an available Job Server. ○ Sub data flow: A resource-intensive operation (such as a sort, table comparison, or table lookup) within a data flow can execute on an available Job Server. Note Casing and spacing are important for these values.

10. Navigate back to the *Web Services Status* page, choose the WSDL version you want to create, and click *View WSDL*.

    A new browser window opens with the WSDL displayed. Use the information in this file to perform the following:

    - Confirm that the software updated the WSDL file with all jobs and services without error.
    - Create calls to the software.
      Use the information in the WSDL file to configure your application to access batch jobs and real-time services.
      To ensure that your application calls the latest version of the job, update the WSDL when the metadata imported into the software changes for a job or real-time service by removing then re-adding a job or service from the *Web Services Configuration* page.

11. After your web service clients are accessing jobs, you can monitor the status of web service operations on the server by viewing the data on the *Web Services Status* page.

Column name	Description
Operation Name	Same as job or real-time service name.
Web Services Port	For jobs, the port name is Batch_jobs. For services, the port name is Real-Time_Service. For built-in operations, the port name is Connection_Operations. For administrative operations for batch jobs, the port name is Batch_Job_Admin.
Repository/ Access Server	For jobs, the repository name. For services, the Access Server name.
Job Information	For jobs, a link to the Batch Job History page. For services, a link to the Real-time Services History page.

Column name	Description
Requests Processed	Number of client requests successfully processed.
Requests Failed	Number of client requests that failed somewhere between the time that the Web Server receives the request and the Job Server receives it.
Requests Pending	Number of requests in a queue for Job Server.
Jobs Failed	Number of requests that failed due to a problem with the Job Server.

7.3.1.2 Tips for using the WSDL file

The WSDL file:

- Appears in the View WSDL window or any browser window by searching for the following URL: http://**<hostname:port>**/DataServices/servlet/webservices?ver=2.0&wsdlxml

  > Note
  >
  > To support previously-created datastores using a WSDL file with XML schema simple types, manually delete "ver=2.0&" from the default URL of Web Service and Apply to save as follows:

  http://**<hostname:port>**/DataServices/servlet/webservices?wsdlxml
- Displays all real-time services and jobs enabled for web services in the Administrator.
- Only displays log on, log off, and session ID information when security is enabled.
- Displays XML Schema formats in the *types* element.

7.3.1.3 WSDL versions

From time to time, the WSDL version used by SAP Data Services may change for a variety of reasons. For example, the syntax may change in order to operate more efficiently, or to add support for new technologies.

Although we typically maintain backward compatibility between WSDL versions, we recommend that you move to the newest WSDL version available in your Data Services installation. The latest version often includes improvements in web service execution, and older versions may be deprecated and no longer supported over time.

For more information about the changes between specific WSDL versions, see the *Upgrade Guide*.

Version history

In general, a WSDL version is deprecated when the version of Data Services in which it was introduced is no longer supported.

WSDL Version	Introduced	Deprecated
1.0	Original WSDL release	n/a
1.1	Data Integrator 11.5.0.0	n/a
2.0	Data Integrator 11.7.0.0	n/a
2.1	Data Services 12.2.1	n/a

7.3.2 Creating a client to use web services

To use a published web service, you must know the URL of the target WSDL. The Administrator produces a WSDL file with this URL: `http://<hostname:port>/DataServices/servlet/webservices?ver=2.0&wsdlxml`

The batch or real-time jobs must have previously been exposed as web services.

This section discusses general steps for using a published SAP Data Services web service. The tools you use to develop your web services client are your choice and the exact steps in using those tools vary, but these basic steps apply as a simple overview to all development projects for web services clients.

1. Import the software's WSDL into your development environment to create a web services client application.
   The incorporated web services appear in the hierarchy of your development environment.
2. Open the web service.
   Each available port for the web service is made visible in the IDE.
3. Write the code to call any of the jobs or services provided by the ports.
4. Run the project to execute the code.
   Executing the code initiates the web services job. A connection is made to the web services tier of the Access Server.

The Access Server then sends information to various job servers, which then executes the al_engine process to run the job, and results are sent back to the Web services client application.

7.3.2.1 Design choices

SAP Data Services provides different ways that you can call jobs using web services, each with benefits and drawbacks:

- Individually published job-specific web services (Batch_Jobs and Real-time_Services ports)
  These web services have their schema published directly in the WSDL, and web service development tools can automatically create classes that serialize and deserialize the input and output XML messages. However, you need to create a separate operation for each published job.
- Generalized web services (Batch_Jobs_Admin and Realtime_Service_Admin ports)

These web services take a job name as input, but do not directly expose the input schema of the job. This allows an application to dynamically call different jobs with one web service, but the schema must be known in advance or generated dynamically with another web service call.

7.3.3 Supported web service operations

SAP Data Services creates a WSDL file with a single service definition. It is possible to create multiple service definitions in a WSDL, but many web service implementations do not support more than one service definition. To avoid that limitation, the software creates only one service.

Within the service definition, the software defines ports for:

- `Connection_Operations`
- `Batch_Job_Admin`
- `Real-time_Services`
- `Batch_Jobs`
- `Repo_Operations`

7.3.3.1 Connection port

SAP Data Services generates WSDL that defines connection operations that belong to web services. The software supports the following Connection operations.

Operation	Description
Ping	Verifies the connection to web services
Logon	Verifies secure access before establishing a session
Logout	Terminates a session

Note

The software generates Logon and Logout operations only if you enable security for published jobs.

7.3.3.1.1 Ping

The Ping operation is an empty input message with a ping operation request. The output message is a text string that returns the current SAP Data Services version, which indicates that a connection has been established.

7.3.3.1.2 Logon

The Logon operation is required when you enable SAP Data Services to provide secure communication. To access web services, provide an Administrator login name and password (with Administrator-level privileges). When the software validates them, the logon operation returns an Administrator session ID that you must include in all subsequent calls to the web services.

Input message

Element	Type	Description
cms_authentication	String	Specifies the type of authentication to use for logging on to a Data Services web service. Values include: • `secEnterprise`: Use for Enterprise authentication • `secLDAP`: Use for LDAP authentication • `secWinAD`: Use for Windows Active Directory authentication • `secSAPR3`: Use for SAP authentication
cms_system	String	The CMS server name. If the CMS server is listening on the default 6400 port, then pass only the CMS server name. If the server is listening on any other port, then also pass the port number.
password	String	Password for the CMS user.
username	String	CMS user name.

Output message

Element	Type	Description
SessionId	String	A unique session ID is returned. Use this session ID in subsequent Data Services operations for which session security is enabled.

7.3.3.1.3 Logout

The Logout operation is required when you enable SAP Data Services to provide secure communication. When web service communication is complete, call the Logout operation to terminate the session.

7.3.3.2 Realtime_Service_Admin port

7.3.3.2.1 Get_RTMsg_Format

Use Get_RTMsg_Format to retrieve the input/output format for a real-time service as an XML Schema. The real-time service does not need to be published as a web service.

Input message

Element	Type	Description
serviceName	string	The name of the real-time service as displayed in the Administrator.
selector	string	A selector that determines whether the input or output schema for the service is returned. Valid strings include: • in - Returns the input schema. • out - Returns the output schema.

Output message

Element	Type	Description
schema	string	The input or output XML Schema for the real-time service.
rootElement	string	The root element of the returned XML Schema.
rootElementNS	string	The root element namespace of the returned XML Schema.
schemaName	string	The name of a dependent schema used in the returned XML Schema, if applicable. This element may be returned multiple times.
schema	string	A dependent schema used in the returned XML Schema, if applicable. This element may be returned multiple times.
errorMessage	string	Any error message that resulted while retrieving the XML Schema for the real-time service.

7.3.3.2.2 Get_RTService_List

Use the Get_RTService_List operation to retrieve a list of the names of published real-time services.

Input message

Get_RTService_List takes no input message.

Output message

Element	Type	Description
serviceName	string	The list of published real-time services.

Element	Type	Description
errorMessage	string	Any error message that occurred while retrieving the list of real-time services.

7.3.3.2.3 Run_Realtime_Service

Use Run_Realtime_Service to call a published real-time service. The real-time service must be running and published as a web service in the Administrator, and the XML input content must match the input format defined for the real-time service.

Input message

Element	Type	Description
serviceName	string	The name of the real-time service as displayed in the Administrator.
xmlInput	string	The XML input content used to start the real-time service. This content must match the input format required by the real-time service.

Output message

Element	Type	Description
xmlOutput	string	The XML output content returned by the real-time service. This content is formatted according to the output schema of the real-time service called.
errorMessage	string	Any error message that resulted while attempting to call the real-time service.

7.3.3.3 Batch_Job_Admin port

7.3.3.3.1 Get_BatchJob_ExeDetail

Use Get_BatchJob_ExeDetail to retrieve a list of all job executions for the selected repository and job. You can also filter the list by time range.

> Note
>
> When the StartTime and EndTime elements are empty, this operation returns all the batch job details. When one field is empty, this operation returns the current date as the default value.

Input message

Element	Type	Description
jobName	String	The name of the batch job.
repoName	String	The name of the repository.
startTime	String	The date the job started (**<YYYY-MM-DD HH:mm:ss>**). For example, `2013-06-08 10:10:10`. This element is optional. When it is empty, the software returns the current date as default value.
endTime	String	The date the job stopped (**<YYYY-MM-DD HH:mm:ss>**). For example, `2013-06-08 10:10:10`. This element is optional. When it is empty, the software returns the current date as default value.

Output message

Element	Type	Description
ObjID	String	The job's unique identifier.
RunID	String	The unique ID for the batch job run.
JobName	String	The name of the batch job.
StartTime	String	The date and time the job started (**<yyyy-MM-dd HH:mm:ss am>**).
EndTime	String	The time the job stopped (**<yyyy-MM-dd HH:mm:ss am>**).
ExecutionTime	String	The length of time (in seconds) the job ran.
Status	String	The status code for the batch job run. Valid strings include: ● Succeed ● Recovered ● Failed
JobServerUsed	String	The name of the job server and the port number. The format is **<jobserver name:jobserver port>**.

Example

SOAP request XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.businessobjects.com/DataServices/ServerX.xsd">
   <soapenv:Header>
      <ser:session>
         <SessionID>D19C201F-074C-B5FE-70A4-2F3436AC04E8</SessionID>
      </ser:session>
   </soapenv:Header>
   <soapenv:Body>
      <ser:jobExcutionDetailsRequest>
         <jobName>Job1_Query</jobName>
         <repoName>echo_oracle</repoName>
         <!--Optional:-->
         <startTime>2013-06-18</startTime>
         <!--Optional:-->
         <endTime>2013-06-19</endTime>
      </ser:jobExcutionDetailsRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Example

SOAP response XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <localtypes:jobExcutionDetailsResponse xmlns:localtypes="http://
www.businessobjects.com/DataServices/ServerX.xsd">
         <jobDetail>
            <ObjID>146</ObjID>
            <RunID>37</RunID>
            <JobName>Job1_Query</JobName>
            <StartTime>2013-06-18 12:22:41 PM</StartTime>
            <EndTime>2013-06-18 12:22:55 PM</EndTime>
            <ExecutionTime>14</ExecutionTime>
            <Status>error</Status>
            <JobServerUsed>echo_orc:3500</JobServerUsed>
         </jobDetail>
         <jobDetail>
            <ObjID>146</ObjID>
            <RunID>38</RunID>
            <JobName>Job1_Query</JobName>
            <StartTime>2013-06-18 12:23:18 PM</StartTime>
            <EndTime>2013-06-18 12:23:32 PM</EndTime>
            <ExecutionTime>14</ExecutionTime>
            <Status>succeeded</Status>
            <JobServerUsed>echo_orc:3500</JobServerUsed>
         </jobDetail>
         <StatusCode>0</StatusCode>
      </localtypes:jobExcutionDetailsResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

7.3.3.3.2 Get_BatchJob_FlowDetails

Use Get_BatchJob_FlowDetails to get details about the tasks included in a job.

Input message

Element	Type	Description
repoName	String	The name of the repository to access.
runID	String	The unique ID for the batch job.

Output message

Element	Type	Description
ObjectName	String	The name of the object used in the job.
ObjectType	String	The type of object. For example, workflow, dataflow, and so on.
ParentObject	String	The name of the main or base object.
ParentType	String	The type of parent object. For example, workflow, dataflow, and so on.
StartTime	String	The date and time that the job execution instance started (**<yyyy-MM-dd HH:mm:ss am/pm>**). For example, 2013-06-18 10:05:38 AM.
EndTime	String	The date and time that this job execution instance stopped (**<yyyy-MM-dd HH:mm:ss am/pm>**). For example, 2013-06-18 12:05:38 PM.
Duration	Integer	The time (in seconds) that the job took to complete.
RowsRead	Integer	The number of rows read.
JobserverUsed	String	The name of the job server used to run the job. The format is **<jobserver name:jobserver port>**.
hasAuditData	String	Specifies if there is audit data. Valid strings include: • Yes • No

Example

SOAP request XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.businessobjects.com/DataServices/ServerX.xsd">
   <soapenv:Header>
      <ser:session>
         <SessionID>D19C201F-074C-B5FE-70A4-2F3436AC04E8</SessionID>
      </ser:session>
   </soapenv:Header>
   <soapenv:Body>
      <ser:jobFlowDetailsRequest>
         <repoName>echo_oracle</repoName>
         <runID>29</runID>
      </ser:jobFlowDetailsRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Example

SOAP response XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <localtypes:jobExcutionDetailsResponse xmlns:localtypes="http://
www.businessobjects.com/DataServices/ServerX.xsd">
         <flowDetail>
            <ObjectName>New_DataFlow10</ObjectName>
            <ObjectType>Dataflow</ObjectType>
            <ParentObject>audit008</ParentObject>
            <ParentType>Job</ParentType>
            <StartTime>2013-06-18 12:05:38 PM</StartTime>
            <EndTime>2013-06-18 12:05:38 PM</EndTime>
            <Duration>0</Duration>
            <RowsRead>0</RowsRead>
            <JobserverUsed>echo_orc:3500</JobserverUsed>
            <hasAuditData>yes</hasAuditData>
         </flowDetail>
         <StatusCode>0</StatusCode>
      </localtypes:jobExcutionDetailsResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

7.3.3.3.3 Get_BatchJob_List

Use Get_BatchJob_List to retrieve a list of the names of published batch jobs.

Input message

Element	Type	Description
repoName	String	The name of the repository to access. This element is optional.
allBatchJobs	Boolean	Includes all batch jobs in the repository, not only those published as web services. This parameter is optional.

Output message

Element	Type	Description
jobName	String	The list of published batch jobs. This element has an additional attribute, repo, which specifies the name of the repository that contains the job.
errorMessage	String	Any error message that occurred while retrieving the list of batch jobs.

7.3.3.3.4 Get_BatchJob_RunIDs

Each individual run of an SAP Data Services batch job is assigned a unique run ID.

Use Get_BatchJob_RunIDs to retrieve a list of run IDs associated with a particular batch job.

Input message

Element	Type	Description
jobName	String	The name of the batch job.
status	String	The status code for the type of run IDs requested. Valid codes include: running, succeeded, error, warning, and all.
repoName	String	The name of the repository to access. When specified, the operation returns only run IDs from this repository. This element is optional.

Output message

The response of the Get_BatchJobs_RunIDs operation contains one or more run element. Each run element contains the following sub-elements:

Element	Type	Description
runID	Integer	The unique ID for the batch job run.
status	String	The status code for the batch job run. Valid codes include: running, succeeded, error, warning, and all.
repoName	String	The repository name associated with the batch job.
errorMessage	String	Any error message that occurred while retrieving the list of batch jobs.

7.3.3.3.5 Get_BatchJob_Status

Use Get_BatchJob_Status to retrieve the status of a particular batch job run.

Input message

Element	Type	Description
runID	Integer	The run ID for the particular batch job status desired.
repoName	String	The name of the repository to access.

Output message

Element	Type	Description
returnCode	Integer	The status for the operation. Valid values include: ● 0 - The operation completed successfully. ● 1 - The operation encountered an error. For example, the repoName specified is invalid.
status	String	The status of the batch job run. Valid values include: ● Running - The job is currently running. ● Succeeded - The job completed with no errors. ● Warning - The job completed, but warnings occurred. ● Error - The job completed with an error.

7.3.3.3.6 Get_DF_Auditdata

Use Get_DF_Auditdata to show audit information for a data flow. To retrieve audit information, you need to enable the audit point in your data flow object.

Input message

Element	Type	Description
runID	String	The unique ID for the batch job.
repoName	String	The name of the repository to access.
dataflow	String	The name of the data flow for which you want audit information.

Output message

Element	Type	Description
auditPoint	String	The name of the object in a data flow from which you are collecting audit statistics.
auditValue	String	The value of the audit function in the audit point.

Example

SOAP request XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.businessobjects.com/DataServices/ServerX.xsd">
   <soapenv:Header>
      <ser:session>
         <SessionID>D19C201F-074C-B5FE-70A4-2F3436AC04E8</SessionID>
      </ser:session>
   </soapenv:Header>
   <soapenv:Body>
      <ser:dataflowAuditRequest>
         <runID>34</runID>
         <repoName>echo_oracle</repoName>
         <dataflow>New_DataFlow10</dataflow>
      </ser:dataflowAuditRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Example

SOAP response XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <localtypes:dataflowAuditResponse xmlns:localtypes="http://
www.businessobjects.com/DataServices/ServerX.xsd">
         <audit>
            <auditPoint>$Count_Query_1</auditPoint>
            <auditValue>6</auditValue>
         </audit>
         <audit>
            <auditPoint>$CountError_Query</auditPoint>
            <auditValue>0</auditValue>
         </audit>
         <audit>
            <auditPoint>$Count_Query</auditPoint>
            <auditValue>6</auditValue>
         </audit>
         <audit>
            <auditPoint>$CountError_Row_Generation</auditPoint>
            <auditValue>0</auditValue>
         </audit>
         <StatusCode>0</StatusCode>
      </localtypes:dataflowAuditResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

7.3.3.3.7 Get_DF_Monitor_Log

Use Get_DF_Monitor_Log to show the runtime statistics for a single data flow execution.

> i Note
>
> The Job Server must be running in order to use this operation.

Input message

Element	Type	Description
repoName	String	The name of the repository to access.
runID	Integer	The unique ID for the batch job.
dataflow	String	The name of the dataflow which contains the batch job.
stoponly	String	Indicates if the process ended without error. You can use a Yes or No string for this element. If it is set to Yes, the output will have only Stop rows, meaning the transform object status is in a Stop state.

Output message

Element	Type	Description
threadName	String	The name of the thread.
state	String	The status of each process in the execution of the data flow. Valid strings include: • Initializing - Indicates that the job is initializing. • Optimizing - Indicates that the job is optimizing. • Proceed - Indicates that the process is executing. • Stop - Indicates that the process ended without error.
absoluteTime	Integer	The amount of time (in seconds) since the execution of the data flow began.

Element	Type	Description
counter	Integer	The number given to each monitor log file to which the software writes information.
rowProcessed	Long	The number of rows processed through the object.
bufferSize	Long	The output buffer size (in kilobytes). This is the amount of time allotted for processing.
bufferUsed	String	The output buffer size used.
CPUUtilization	String	The percent of CPU uses per thread.
jobServerUsed	String	The name of the job server used to execute the data flow.

Example

SOAP request XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.businessobjects.com/DataServices/ServerX.xsd">
   <soapenv:Header>
      <ser:session>
         <SessionID>D19C201F-074C-B5FE-70A4-2F3436AC04E8</SessionID>
      </ser:session>
   </soapenv:Header>
   <soapenv:Body>
      <ser:DataFlowMonitorLogRequest>
         <runID>29</runID>
         <repoName>echo_oracle</repoName>
         <dataflow>New_DataFlow10</dataflow>
         <stoponly>yes</stoponly>
      </ser:DataFlowMonitorLogRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Example

SOAP response XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <localtypes:DataFlowMonitorLogResponse xmlns:localtypes="http://
www.businessobjects.com/DataServices/ServerX.xsd">
         <Row>
            <threadName>/New_DataFlow10/Row_Generation</threadName>
            <state>STOP</state>
            <absoluteTime>2.906</absoluteTime>
            <counter>1</counter>
            <rowProcessed>6</rowProcessed>
            <bufferSize>0</bufferSize>
            <bufferUsed>0</bufferUsed>
            <CPUUtilization>0.0</CPUUtilization>
            <jobServerUsed>echo_orc:3500</jobServerUsed>
         </Row>
         <Row>
            <threadName>/New_DataFlow10/Query-Mapping1</threadName>
            <state>STOP</state>
            <absoluteTime>2.906</absoluteTime>
            <counter>1</counter>
            <rowProcessed>0</rowProcessed>
            <bufferSize>0</bufferSize>
            <bufferUsed>0</bufferUsed>
            <CPUUtilization>0.0</CPUUtilization>
            <jobServerUsed>echo_orc:3500</jobServerUsed>
         </Row>
         <Row>
            <threadName>/New_DataFlow10/Audit[3]</threadName>
            <state>STOP</state>
            <absoluteTime>2.906</absoluteTime>
            <counter>1</counter>
            <rowProcessed>0</rowProcessed>
            <bufferSize>0</bufferSize>
            <bufferUsed>0</bufferUsed>
            <CPUUtilization>0.0</CPUUtilization>
            <jobServerUsed>echo_orc:3500</jobServerUsed>
         </Row>
         <Row>
            <threadName>/New_DataFlow10/Merge: 0</threadName>
            <state>STOP</state>
            <absoluteTime>2.906</absoluteTime>
            <counter>1</counter>
            <rowProcessed>6</rowProcessed>
            <bufferSize>0</bufferSize>
            <bufferUsed>0</bufferUsed>
            <CPUUtilization>0.0</CPUUtilization>
            <jobServerUsed>echo_orc:3500</jobServerUsed>
         </Row>
         <Row>
            <threadName>-New_DataFlow10/output8.txt</threadName>
            <state>STOP</state>
            <absoluteTime>2.906</absoluteTime>
            <counter>1</counter>
            <rowProcessed>6</rowProcessed>
            <bufferSize>0</bufferSize>
            <bufferUsed>0</bufferUsed>
            <CPUUtilization>0.0</CPUUtilization>
            <jobServerUsed>echo_orc:3500</jobServerUsed>
         </Row>
         <returnCode>0</returnCode>
      </localtypes:DataFlowMonitorLogResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

7.3.3.3.8 Get_DF_Monitor_Log_Pivot

Use Get_DF_Monitor_Log_Pivot to show the runtime statistics as pivoted for a single data flow execution.

i Note

The Job Server must be running in order to use this operation.

Input message

Element	Type	Description
repoName	String	The name of the repository to access.
runID	Integer	The unique ID for the batch job.
dataflow	String	The name of the data flow for which you want audit information.
threadName	String	Filters the output based on the thread name. This element is optional.
measure	String	Filters the output based on the measure type. This element is optional. Valid strings include: ● rows processed ● buffer used ● thread CPU %

Output message

Element	Type	Description
threadName	String	The name of the thread.
measure	String	The measure type. Valid strings include: ● rows processed ● buffer used ● thread CPU %
sample<x>	Integer	The software writes information to log files at assigned intervals. The sample number corresponds to the monitor sample counter, which is a number that is assigned to each monitor log file. The sample output consists of measure type statistics for a thread. Output is limited to 500 sample columns (sample1 through sample500).

Example

SOAP request XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.businessobjects.com/DataServices/ServerX.xsd">
   <soapenv:Header>
      <ser:session>
         <SessionID>D19C201F-074C-B5FE-70A4-2F3436AC04E8</SessionID>
      </ser:session>
   </soapenv:Header>
   <soapenv:Body>
      <ser:DataFlowMonitorLogPivotRequest>
```

```
         <runID>29</runID>
         <repoName>echo_oracle</repoName>
         <dataflow>New_DataFlow10</dataflow>
         <!--Optional:-->
         <threadName></threadName>
         <!--Optional:-->
         <measure>thread CPU %</measure>
      </ser:DataFlowMonitorLogPivotRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Example

SOAP response XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <localtypes:DataFlowMonitorLogPivotResponse xmlns:localtypes="http://
www.businessobjects.com/DataServices/ServerX.xsd">
         <Row>
            <threadName>/New_DataFlow10/Row_Generation</threadName>
            <measure>thread CPU %</measure>
            <Sample1>0.0</Sample1>
            <Sample2>0.0</Sample2>
            <Sample3>0.0</Sample3>
            <Sample4/>
            <Sample5/>
            <Sample6/>
            <Sample7/>
            <Sample8/>
            <Sample9/>
            <Sample10/>
            <Sample11/>
                                                ...
                                                <Sample496/>
            <Sample497/>
            <Sample498/>
            <Sample499/>
            <Sample500/>
         </Row>
         <returnCode>0</returnCode>
      </localtypes:DataFlowMonitorLogPivotResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

7.3.3.3.9 Get_Error_Log

SAP Data Services produces several types of log information for a batch job published as a web service.

Use the Get_Error_Log operation to retrieve the error log for a batch job.

Input message

Element	Type	Description
runID	integer	The batch job run ID for the particular log desired.
repoName	string	The name of the repository to access.
page	integer	The page number of the error log to return. This element is optional.

Output message

Element	Type	Description
returnCode	integer	The status for the operation. Valid values include: • 0 - The operation completed successfully. • 1 - The operation failed to retrieve the error log.
error	string	The error log associated with the input batch job run ID.

7.3.3.3.10 Get_Job_Input_Format

Use Get_Job_Input_Format to retrieve the input format for a batch job as an XML Schema.

Input message

Element	Type	Description
jobName	string	The name of the batch job as displayed in the Administrator.
repoName	string	The name of the repository to access.

Output message

Element	Type	Description
format	string	The input format for the batch job, as an XML Schema.
errorMessage	string	Any error message that resulted while retrieving the input format for the batch job.

7.3.3.3.11 Get_Monitor_Log

SAP Data Services produces several types of log information for a batch job published as a web service.

Use the Get_Monitor_Log operation to retrieve the monitor log for a batch job.

Input message

Element	Type	Description
runID	integer	The batch job run ID for the particular log desired.
repoName	string	The name of the repository to access.
page	integer	The page number of the monitor log to return. This element is optional.

Output message

Element	Type	Description
returnCode	integer	The status for the operation. Valid values include: • 0 - The operation completed successfully. • 1 - The operation failed to retrieve the monitor log.
monitor	string	The monitor log associated with the input batch job run ID.

7.3.3.3.12 Get_Trace_Log

SAP Data Services produces several types of log information for a batch job published as a web service.

Use the Get_Trace_Log operation to retrieve the trace log for a batch job.

Input message

Element	Type	Description
runID	integer	The batch job run ID for the particular log desired.
repoName	string	The name of the repository to access.
page	integer	The page number of the trace log to return. This element is optional.

Output message

Element	Type	Description
returnCode	integer	The status for the operation. Valid values include: • 0 - The operation completed successfully. • 1 - The operation failed to retrieve the trace log.
trace	string	The trace log associated with the input batch job run ID.

7.3.3.3.13 Run_Batch_Job

Use Run_Batch_Job to call a batch job with the ability to specify job parameters and global variables.

Input message

Element	Type	Description
jobName	string	The name of the batch job as displayed in the Administrator.
repoName	string	The name of the repository to access.
jobServer	string	The name of the job server to use to execute the job. This element is optional, but cannot be specified if serverGroup is also specified.
serverGroup	string	The name of the server group to use to execute the job. This element is optional, but cannot be specified if jobServer is also specified.
jobParameters	complex	A complex XML element that sets specific job execution parameters.
globalVariables	complex	A complex XML element that defines global job execution variables.

> Note
>
> If you do not specify a serverGroup or jobServer, the operation attempts to find an available job server that is attached to the repository by first checking the server group list, and then the job server list.

> Note
>
> For detailed information about the jobParameters and globalVariables elements, view the WSDL from the Administrator. For more information about available job parameters and global variables, see the *Reference Guide*.

Output message

Element	Type	Description
pid	int	The process ID number for the batch job execution. Process IDs can be reused.
cid	int	The counter ID number for the batch job excecution. You can use a counter ID together with a process ID to uniquely identify a specific instance of the job execution.
rid	int	The run ID associated with the specific execution of the batch job.
repoName	string	The repository name associated with the batch job execution.
errorMessage	string	Any error message that resulted while attempting to call the batch job.

7.3.3.3.14 Stop_Batch_Job

Use the Stop_Batch_Job operation to stop a particular batch job run.

Input message

Element	Type	Description
runID	integer	The run ID for the particular batch job run to stop.
repoName	string	The name of the repository to access.

Output message

Element	Type	Description
returnCode	integer	The success code for the batch job stop attempt. Valid codes include: • 0 - The operation successfully stopped the specified batch job. • 1 - The operation failed to stop the specified batch job in the specified repository.
errorMessage	string	The error message associated with a failure to stop a batch job run.

7.3.3.4 Real-time_Services port

SAP Data Services generates WSDL that defines how to invoke real-time services enabled as web service operations. Each real-time service name is represented as an operation name in the WSDL file.

Each real-time service operation has a set of messages that it uses to communicate with the real-time service. Real-time services use a defined XML message as input and a defined XML message as output. Real-time services obtain the format of these messages from the repository and provide the format in the WSDL.

The software supports XML Schemas as its message format for real-time services. A web service provides only XML Schemas in the WSDL. You will need to convert any DTDs to XML schemas as necessary.

The repository stores XML Schemas that define the input and output messages as independent definitions. The WSDL file includes these definitions in the *types* element.

The messages that an XML Schema defines for each real-time service operation are:

- Header message
  If security is enabled for the message, the software defines a secure session identifier in the message header.
- Input message
  When an external web services client invokes it, the input message passes information to a real-time service. The name of the input message is the name of the operation that the software publishes followed by the suffix `_Input`. The input message contains the message source defined by the real-time service.
- Output message
  The software returns the output message when the real-time service completes. The output message contains the output generated by the real-time service. The name of the output message is the name of the operation followed by the suffix `_Output`. The output message contains the message target defined by the real-time service.
- Fault message
  The software returns a fault message when it cannot invoke the real-time service.

7.3.3.4.1 Message formats

The following segment shows the syntax that Data Services generates in a WSDL file to define an operation's messages. In this example:

- `RTService` represents the name of the real-time service as defined in the Administrator.
- `XMLSchemaName` represents the name of the XML Schema that was used to create an XML message source or XML message target in the Designer.
- `RootElement` represents the root element of the XML Schema.

The software publishes a WSDL that includes input and output XML Schema message formats in the *types* element.

> **i Note**
>
> Server support for real-time services requires that you use a valid URL for locating XML Schema (.xsd) files in an import statement. A local file name cannot be used. For example, the .xsd must be either self-contained when imported into the software or it must use a network reference (URL), not a file or relative path reference, as an identifier.

The WSDL file displays the operations for real-time services within a `portType` tag.

7.3.3.5 Batch_Jobs port

SAP Data Services generates WSDL that defines how to start batch jobs. The WSDL file represents each batch job name as an operation.

In addition, the WSDL file defines an input and output message for each operation. An input message communicates the input needed by the job at startup (such as the global variables needed to start the job). An output message either confirms that the job started or presents the errors that prevent the job from starting.

WSDL defines the following messages for each operation:

Message	Description
Header message	When security is enabled for the message, the software defines a secure session identifier in the message header.
Input message	The input message passes information needed by the batch job at startup. The name assigned to the input message is the name of the operation followed by the suffix `_Input`. The input message contains global variables. When security is enabled for the message, the software defines a secure session identifier in the message header.
Output message	The software returns the output message when the batch job starts. The output message contains the job identification. The name of the output message is the name of the operation followed by the suffix `_Output`. The output message contains the following IDs: ● OS process ID of the started job ● Job Server Counter ID of the started job
Fault message	The software returns a fault message if the batch job fails to start. It returns a text description of the error that prevents the job from starting.

7.3.3.5.1 SoapAction element

The definition of each batch job operation uses the `soapAction` element to define the batch job name needed to launch the job.

The WSDL file displays the `soapAction` element in the service and port section.

7.3.3.5.2 Security

When publishing a job as a web service, the Administrator can enable secure access, requiring that web services clients provide authentication and authorization (an Administrator username and password) for access to the web service operations. Administrator-level (not Monitor-level) privileges must be used. That is, you cannot limit access to users based on role. This authentication is SSL-compliant.

If you do not enable secure access and you are using HTTP, web services clients have open access to all published batch jobs and real-time services.

Related Information

To configure web service information using the Administrator [page 2221]

7.3.3.6 Repo_Operations port

SAP Data Services generates WSDL that defines operations that belong to web services. The software supports the following operations on the Repo_Operations port.

Operation	Description
Delete_Repo_Objects	Deletes objects from the repository.
Import_Repo_Object	Imports an object to the repository.
Validate_Repo_Object	Validates an object contained in the repository.
Export_DQReport	Exports reports to a specified location at runtime.
Get_Repository_List	Retrieves the entire repository list based on the user's permissions. This operation also returns a list of the repositories that the user is currently logged into.

7.3.3.6.1 Delete_Repo_Objects

Use the Delete_Repo_Objects operation to delete objects from the SAP Data Services repository.

Input message

Element	Type	Description
objName	string	The name of the object to delete from the repository. This element requires the attribute objType and can occur multiple times. The objType attribute specifies the type of the object: • BATCH_JOB • REALTIME_JOB • WORKFLOW • DATAFLOW

Element	Type	Description
		- ABAP_DATAFLOW - DATA_QUALITY_TRANSFORM_CONFIGURATION - DATASTORE - FILE_FORMAT - XML_SCHEMA - DTD - CUSTOM_FUNCTION - EXCEL_WORKBOOK - COBOL_COPYBOOK - SYSTEM_PROFILE - SUBSITUTION_CONFIGURATION - PROJECT - TABLE - TEMPLATE_TABLE - DOMAIN - HIERARCHY - STORED_PROCEDURE - IDOC - BW_MASTER_TRANSFER_STRUCTURES - BW_MASTER_TEXT_TRANSFER_STRUCTURES - BW_TRANSACTION_TRANSFER_STRUCTURES - BW_HIERARCHY_TRANSFER
repoName	string	The name of the repository that contains the objects to delete.
jobServer	string	The name of the job server associated with the repository. This element is optional, but cannot be specified if serverGroup is also specified.
serverGroup	string	The name of the server group associated with the repository. This element is optional, but cannot be specified if jobServer is also specified.
traceOn	string	Enables tracing for the operation. This element is optional.

Note

If you do not specify a serverGroup or jobServer, the operation attempts to find an available job server that is attached to the repository by first checking the server group list, and then the job server list.

Output message

Element	Type	Description
returnCode	int	The status of the operation: - 0 - The operation completed successfully.

Element	Type	Description
		- 1 - The operation failed to complete successfully.
errorMessage	string	The error message associated with the operation. This element is output only if the operation fails to complete successfully.
traceMessage	string	The trace message associated with the operation. This element is output only if the traceOn element is specified on input.

7.3.3.6.2 Import_Repo_Object

Use the Import_Repo_Object operation to save an XML object definition to the SAP Data Services repository.

Input message

Element	Type	Description
definition	string	The object to import to the repository. The object must be defined in XML format.
repoName	string	The name of the repository in which to import the object.
jobServer	string	The name of the job server associated with the repository. This element is optional, but cannot be specified if serverGroup is also specified.
serverGroup	string	The name of the server group associated with the repository. This element is optional, but cannot be specified if jobServer is also specified.
traceOn	string	Enables tracing for the operation. This element is optional.
passphrase	string	The passphrase that was used to encode any passwords in the XML object definition. Note If the passphrase you specify is incorrect, the operation will still save the XML objects to the repository, but any passwords in the definition will be removed.

> Note
>
> If you do not specify a serverGroup or jobServer, the operation attempts to find an available job server that is attached to the repository by first checking the server group list, and then the job server list.

Output message

Element	Type	Description
returnCode	int	The status of the operation: • 0 - The operation completed successfully. • 1 - The operation failed to complete successfully.
errorMessage	string	The error message associated with the operation. This element is output only if the operation fails to complete successfully.
traceMessage	string	The trace message associated with the operation. This element is output only if the traceOn element is specified on input.

7.3.3.6.3 Validate_Repo_Object

Use the Validate_Repo_Object operation to validate an object stored in the SAP Data Services repository.

Input message

Element	Type	Description
objName	string	The name of the object to validate.
objType	string	The type of the object to validate: • BATCH_JOB • REALTIME_JOB • WORKFLOW • DATAFLOW • ABAP_DATAFLOW • DATA_QUALITY_TRANSFORM_CONFIGURATION • CUSTOM_FUNCTION
repoName	string	The name of the repository that contains the object to validate.
systemProfile	string	The name of the job system profile to use while validating the object. This element is optional.
jobServer	string	The name of the job server associated with the repository. This element is optional, but cannot be specified if serverGroup is also specified.
serverGroup	string	The name of the server group associated with the repository. This element is optional, but cannot be specified if jobServer is also specified.
substitutionParameters	complex	Substitution parameters to override while validating the object. This element is optional and contains one or more parameter child elements.
parameter	string	An individual substitution parameter.
traceOn	string	Enables tracing for the operation. This element is optional.

i Note

If you do not specify a serverGroup or jobServer, the operation attempts to find an available job server that is attached to the repository by first checking the server group list, and then the job server list.

Output message

Element	Type	Description
returnCode	int	The status of the operation: • 0 - The operation completed successfully. • 1 - The operation failed to complete successfully.
errorMessage	string	The error message associated with the operation. This element is output only if the operation fails to complete successfully.
traceMessage	string	The trace message associated with the operation. This element is output only if the traceOn element is specified on input.

7.3.3.6.4 Export_DQReport

Use the Export_DQReport operation to export reports to a specified location at runtime.

Input message

Element	Type	Description
runID	integer	The unique ID for the batch job run.
repoName	string	The name of the repository to that contains the batch job.

Output message

Element	Type	Description
exportFileName	string	The name of the file that is exported; for example, `matchcriteriasummary_Set1.pdf`.
exportPath	string	The path where the reports will be exported to. The default path is <**<DS_COMMON_DIR>**>`\DataQuality\reports\`. Upon execution, the repository name and job name folders are appended to the path. If the Overwrite option is not selected, a run ID folder is also appended to the path.

Element	Type	Description
		Note If you export reports to a location other than a local drive, such as a network drive, before you execute the job you must start the web application server with an account that has access rights to that location.
exportStatus	boolean	The status of the overall export operation: ● 0 - The operation completed successfully. ● Negative integer - The operation failed to complete successfully.
processMessage	string	Informational messages about the overall export process.
reportName	string	The name of the report that is displayed in the Management Console; for example, Match Criteria Summary.
reportStatus	integer	The status of the export operation for each report: ● 0 - The operation completed successfully. ● Negative integer - The operation failed to complete successfully.
statusMessage	string	Informational message about the export status for each report.

7.3.3.6.5 Get_Repository_List

Use Get_Repository_List to retrieve the entire repository list based on the user's permissions. This operation also returns a list of the repositories that the user is currently logged into.

Input message

No input parameter is needed. You only need to log into the system.

Output message

Element	Type	Description
repoName	String	The name of the repository that is configured in the Management Console.
repoType	String	The repository database type. This element is optional. Valid strings include: ● local ● central

Element	Type	Description
		• secure central • profiler
dbType	String	The repository database type. For example, Oracle, DB2, HANA, and so on.
dbHost	String	The host name of the database server.
connectionStatus	String	Indicates whether the connection is active or inactive.
username	String	The user name used to log into the database.
permissions	String	The user's permission level for the repository. If you don't have rights to view a repository, the repository name will not be returned. Valid strings include, READ, EDIT, and DELETE.

Example

SOAP request XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.businessobjects.com/DataServices/ServerX.xsd">
   <soapenv:Header>
      <ser:session>
         <SessionID>D19C201F-074C-B5FE-70A4-2F3436AC04E8</SessionID>
      </ser:session>
   </soapenv:Header>
   <soapenv:Body>
      <ser:GetListOfRepositoriesRequest/>
   </soapenv:Body>
</soapenv:Envelope>
```

Example

SOAP response XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <localtypes:GetListOfRepositoriesResponse xmlns:localtypes="http://
www.businessobjects.com/DataServices/ServerX.xsd">
         <repository>
            <repoName>echo_oracle</repoName>
            <repoType>local</repoType>
            <dbType>Oracle</dbType>
            <dbHost>shg-v-imdb02</dbHost>
            <connectionStatus>active</connectionStatus>
            <username>echo</username>
            <permissions>READ,EDIT,DELETE</permissions>
         </repository>
         <repository>
            <repoName>echo_sqlany</repoName>
            <repoType>local</repoType>
            <dbType>SQL_Anywhere</dbType>
            <dbHost>shg-v-imdb03</dbHost>
            <connectionStatus>active</connectionStatus>
            <username>dba</username>
            <permissions>READ,EDIT,DELETE</permissions>
         </repository>
      </localtypes:GetListOfRepositoriesResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

7.3.4 Enabling SSL support

7.3.4.1 To configure SSL on the web application server

For SAP Data Services web services to work with SSL, the web application server must be configured to support SSL connections. The `server.xml` file can be used to configure the packaged Tomcat application server.

> Note
>
> For other web application servers, refer to the product documentation about how to configure SSL support.

1. Open `server.xml` in a text editor. This file is located in the `Tomcat55\conf` directory at the same level as `LINK_DIR`.
2. Locate the commented `connector` element in the XML:

   ```
   <!-- Define a SSL HTTP/1.1 Connector on port 8443 -->
   <!--
   <Connector port="8443" maxHttpHeaderSize="8192"
     maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
     enableLookups="false" disableUploadTimeout="true"
     acceptCount="100" scheme="https" secure="true"
     clientAuth="false" sslProtocol="TLS" />
    -->
   ```

3. Remove the comment (`<!-- -->`) tags around the `connector` element.
4. Add the `keystoreFile` and `keystorePath` attributes into the `connector` element.

   ```
   keystoreFile="<path/to/keystore/file>"
   keystorePass="<keystore_password>"
   ```

5. Restart the Tomcat application server.

7.3.5 Error reporting

SAP Data Services uses web services to define every operation with both an input and output message. In addition to the output message, the software returns a fault message when an error occurs.

7.3.5.1 Administrator log

In addition to the fault message, SAP Data Services writes log and debug messages to the Administrator's log file (`webadmin.log`). Fault messages include a short description of a failure. For detailed information about an error, use the Administrator's log file.

All Administrator components share the Administrator's log. The software prefixes these messages with the name of the component that issues the error message. For web services, the component name is the name of the Java class issuing the error. All web service classes start with `com.acta.adapter.webservice`.

The software creates the Administrator's log file in: **`<DS_COMMON_DIR>`**`\log\webadmin.log`.

To control the level of detail in the `webadmin.log` file, you must edit the `log4j.properties` file. The properties file is located in:

`<LINK_DIR>``\ext\webserver\webapps\acta_web_admin\WEB-INF`

To obtain a debug trace of events, change the log level from the default of `INFO` to `DEBUG`. For example, `log4j.rootLogger=DEBUG, A`

7.3.5.2 Web service log

In addition to the shared Administrator log, web service messages are also written to a separate log file. The `WebService.log` file is in **`<DS_COMMON_DIR>`**`\log`.

7.3.5.3 Error messages

The following are error messages that you might encounter if you are using SAP Data Services as a web service provider to accept inbound calls:

- A web service is unable to process a request due to an unknown function in the soapAction element.
  The server returns this error message if the soapAction header in the HTTP request is not recognized. Every web service call expects a soapAction header that indicates an action. The WSDL publishes a soapAction for each operation. When the web services server cannot determine what action to take, it is unable to call the software.
  To find extended error information, use the `WebService.log` file in **`<DS_COMMON_DIR>`**`/log`. To use extended diagnostics, use debug tracing in the `webadmin.log` file.
- A web service is unable to process a request to call real-time service **`<ServiceName>`** using Access Server **`<AccessServerName>`**.
  The server returns this error message when it recognizes a request to call a real-time service but is unable to extract the XML message from the SOAP Envelope that is supposed to be sent to the real-time service. To find extended error information, use the `WebService.log` file in **`<DS_COMMON_DIR>`**`/log`.
  To use extended diagnostics, use debug tracing in the `webadmin.log` file.
- A web service sent a request to invoke real-time service **`<ServiceName>`** to Access Server **`<AccessServerName>`**. The request failed with error = **`<access-server generated error message>`**

The server returns this error message if it recognizes a request to call a real-time service, data was extracted from the incoming SOAP Envelope, and data was passed to the Access Server, which refused to service the request.

To locate where the error occurred use the Access Server log file (*Real-time* > **<AccessServerName>** > *Logs - Current*).

If the Access Server passed the request on to the Job Server, use the following logs to diagnose the problem:

  - Job Server log (**<DS_COMMON_DIR>**`/log/`**<JobServerName>**`/server_eventlog.txt`)
  - Real-time service provider log (*Real-time* > **<AccessServerName>** > **<Real-timeServiceName>** > **<ProcessID>**)

- A web service operation is unable to process the request to start batch job **<JobName>** on server **<JobServerName>**. Error = **<web services generated error message>**
  The server returns this error message if it recognizes a request to start a batch job but is unable get the information it needs to start the job.
  To find extended error information, use the `WebService.log` file in **<DS_COMMON_DIR>**`/log`. To use extended diagnostics, use debug tracing in the `webadmin.log` file.
- A web service sent a request to start batch job **<JobName>** on server **<JobServerName>**. The Job Server refused the request with error: **<job server generated error message>**
  The server returns this error message if it recognizes a request to start a batch job and has passed the request to the Job Server to start the job. The Job Server is unable to start the job. To find extended error information, use the Job Server log (**<DS_COMMON_DIR>**`/log/`**<JobServerName>**`/server_eventlog.txt`).

Related Information

Administrator log [page 2256]

7.4 Consuming external web services in SAP Data Services

You can add functionality to SAP Data Services to invoke web services in external applications from data flows. This functionality requires configuring the software's built-in web services datastore type. The web services datastore provides support for locating and importing metadata for a web services server as well as invoking web service operations.

The web services datastore works by sending a request and waiting until it receives a reply from a web services server.

For example, you might create a web services server as a front-end to a legacy application. You could call the web services server daily from a data flow to access inventory and update an inventory data mart.

The interaction between the web services datastore and an external web service has these parts:

- Creating a web services datastore that identifies the WSDL, which describes the web services server.
- Importing metadata to extract the information form the WSDL needed to access the web service server.

- Creating a data flow that uses the imported function call to call the web services server.

7.4.1 To access a web service using the Designer

To configure access to a specific web services, use the Designer to create a web service datastore. SAP Data Services provides access to web services as stream-oriented function calls, which it configures when you import metadata.

1. Create a web service datastore.

   The following parameters are available when creating a Web Service SOAP datastore:

Parameter	Details
Datastore type	Choose Web Service SOAP.
Web Service URL	Specify the URL of the web services server for a data flow to access. It must be the same URL that accepts a web service connection and returns the WSDL. The datastore connects to the web services server using the URL to locate the definition of published services.
User name	Enter the user name for HTTP basic authentication. Required only if basic authentication is needed to connect to the web service provider.
Password	Enter the password for HTTP basic authentication. Required only if basic authentication is needed to connect to the web service provider.
XML Recursion Level	Enter the number of passes the software should run through the XSD to resolve names. The default is 0.
Keystore path	If the web service provider uses an SSL connection, specify the location of the keystore used to establish the connection.
Socket timeout	Enter the maximum number of milliseconds the web service client will wait to receive the response from the web service provider.
Axis2/c config file path	Enter the path to your Axis2/c configuration file (`axis2.xml`). If a path is not specified, the default path is **`<LINK_DIR>`**`/ext/webservice-c/axis2.xml`.
Proxy host	Enter the host name for the HTTP proxy server. Required only if a proxy server is needed to connect to the web service provider.
Proxy port	Enter the port number for the HTTP proxy server. Required only if a proxy server is needed to connect to the web service provider.
Proxy username	Enter the user name for the HTTP proxy server. Required only if a proxy server that uses authentication is needed to connect to the web service provider.
Proxy password	Enter the password for the HTTP proxy server. Required only if a proxy server that uses authentication is needed to connect to the web service provider.

   The following parameters are available, depending on the Authorization Schema you select, when creating a Web Service REST datastore:

Parameter	Details
Datastore type	Choose Web Service REST.
Web Service URL	Specify the location of the WADL. You can enter the URL or the path to a local WADL file.
Authorization schema	Choose one of the following options:

Parameter	Details
	○ Basic ○ Basic + CSRF ○ Authorization Header ○ OAuth 1.0 (2-legged) ○ OAuth 2.0 (2-legged) Data Services does not support the 3-legged model.
User name	Enter the user name for HTTP basic authentication. Required only if basic authentication is needed to connect to the web service provider.
Password	Enter the password for HTTP basic authentication. Required only if basic authentication is needed to connect to the web service provider.
Password Type	Options include Plain and Digest.
CSRF Fetch URL CSRF Fetch URL Method CSRF Header Key CSRF Header Value	Enter the location (URL) to fetch the CSRF token. Tokens can be retrieved via a service call to the server. To retrieve the token, set *CSRF Fetch URL Method* to GET or POST, *CSRF Header Key* (case sensitive) to X-CSRF-Token or the server required header key, and *CSRF Header Value* to Fetch or the server required value.
Header based api key or token	Enter the api key or token. For example, Token token=SECRET.
Proxy host	Enter the host name for the HTTP proxy server. Required only if a proxy server is needed to connect to the web service provider.
Proxy port	Enter the port number for the HTTP proxy server. Required only if a proxy server is needed to connect to the web service provider.
Proxy username	Enter the user name for the HTTP proxy server. Required only if a proxy server that uses authentication is needed to connect to the web service provider.
Proxy password	Enter the password for the HTTP proxy server. Required only if a proxy server that uses authentication is needed to connect to the web service provider.
Standard HTTP Header Fields	Enter a header field. Header fields generally contain the operating parameters of an HTTP request or response.
SSL Pem File	Enter the path and name of the `.pem` file (private key or certificate).
Dynamic Base URL	Enter the base URL. The base URL is comprised of the protocol, server name, port number, and path of the service that listens to RESTful web service requests. You can obtain ths URL from the WADL file under the element resource and attribute base. This is useful when you have multiple servers serving the same resources and you want to dynamically create the URL during the call.
Preferred Mediatype	The following options are available: ○ Application/XML ○ Application/JSON
Consumer Key for Oauth Consumer Key Secret for Oauth	Enter the Consumer Key and Consumer Key Secret for Oauth (equivalent to role account user names and passwords). You can obtain this information through the web service provider (for example, Google Apps or Twitter).

Parameter	Details
Oauth Token Key Oauth Token Key Secret	If the server supports 2-legged OAuth, Data Services does the authentication using the Consumer key and secret, otherwise, enter the Token Key and Token Key Secret for the Oauth. This information allows single user authorization. You can obtain this information through the web service provider (for example, Google Apps or Twitter).
Signature method	Select the signature method for HTTP requests. Options include: ○ HMAC-SHA1 (default) ○ Plain text
Client ID Client Secret	Enter the Client ID (represents your application) and Client Secret (security key). You can obtain this information through the web service provider (for example, Google Apps or Twitter).
Access token	Enter the location (API endpoint) of the temporary token. This allows you to access protected resources.
Refresh token	Enter the Refresh Token.
Request token URL Access token URL	Enter the URL for requesting a temporary token and a URL for getting the final token. Note Leave these parameters empty if the server uses only consumer key and secret to access the protected resource.
Grant type	Enter the type of grant access you want to use to obtain an access token. Options include: ○ Client credentials (default): Use your own credentials in order to obtain an access token. ○ Password: Use the resource owner's password to obtain an access token.
Default configuration	Indicates if you want to use the default configuration (the default is Yes).
Preferred method	Select the method you want to use to test trusted authentication. Options include: ○ Header String (POST) (default) ○ Query String (GET)

2. Import metadata from the web service datastore
   a) From the object library, open the web service datastore.
   The Designer calls the web service server at the indicated WSDL or WADL URL and obtains a list of the published services and ports.
   b) Expand the ports to see published operations available for import.
   c) Right-click an operation and select *Import*.
   The software imports web service operations as function calls and lists them under the web service datastore in the object library. Each function call includes a definition for both the input and output messages required for communication with a web service operation. The Designer extracts the details about the request and reply messages and generates XML Schema that describes the messages.
3. From the Designer, add a web service function call to a job.

   As a web services client, the software calls a web services server twice:

- ○ During design time to import metadata for the functions and data types that a particular web service supports.
- ○ During run time to call the web service and invoke its functionality.

For more information, see “Defining a web service datastore” in the *Designer Guide*.

Related Information

Configuring HTTP header fields [page 2261]

7.4.2 To add web service calls to a job

Once a web service datastore is created and metadata is imported, you can add web service function calls to an SAP Data Services job.

1. Add a Query transform to the data flow.
2. Open the Query editor, right-click the target schema and select *New function call*.
   The Function Editor opens listing the operation metadata that you imported under the datastore name.
3. Select a datastore to view the metadata that you want to add to your job.
4. Select the metadata name and click *Next*.
5. Map the input schema to the output schema.

   > **Note**
   >
   > If you want to nest data in the target schema, use this first Query transform to place the schema in your job and additional Query transforms to perform the nesting. The Function Editor does not allow complex schema configuration.
6. Click *OK*.
   The imported schema appears in the query.
7. Configure the rest of the data flow by supplying input to the function call and extracting the response information obtained from the web service.

7.4.3 Configuring HTTP header fields

In the HyperText Transfer Protocol (HTTP), header fields generally contain the operating parameters of an HTTP request or response. The header fields define various characteristics of the the data transfer that is requested or the data that is provided in the message body.

An HTTP header field always starts with a field name, then a colon (`:`), and ends with the field value. A core set of header fields is standardized by the Internet Engineering Task force. These standard headers are commonly understood by all compliant protocol implementations. Header field names may also be any application-specific strings, known as customary headers.

7.4.3.1 To configure standard HTTP header fields

Standard HTTP header fields have specified values that are the same and fixed for all web service functions in the web service datastore. The values for standard fields also remain the same for all web service calls in a data flow.

1. In the Designer, open the web service datastore configuration and click *Advanced*.
2. Select *Standard HTTP header fields* and click *...*.
   The *Edit HTTP Header Fields* screen appears.
3. Enter each field name and its corresponding default value and click *OK*.
   A semicolon-separated list of the header fields appears in the column for *Standard HTTP header fields*.
4. Click *OK* to accept the datastore configuration.

7.4.3.2 To configure dynamic HTTP header fields

Dynamic HTTP header fields have values that may be different for each function in the web service datastore. The values for dynamic fields can also change for each web service call in a data flow.

1. In the Designer, open the web service datastore configuration and click *Advanced*.
2. Select *Dynamic HTTP header fields* and click *...*.
   The *Edit HTTP Header Fields* screen appears.
3. Enter each field name and the maximum length for its value and click *OK*.
   A semicolon-separated list of the header fields appears in the column for *Dynamic HTTP header fields*.
4. Click *OK* to accept the datastore configuration.

When you import a function into the web service datastore, the dynamic HTTP headers are available for mapping in the HTTPHeader schema.

7.4.4 Enabling SSL support

7.4.4.1 To configure SSL on the native web service datastore

To configure SSL support on the native web service datastore, add the path to your keystore to the datastore configuration.

> Note
>
> The keystore path is only used while importing WSDL operations into the datastore, and is not used at runtime.

For more information about configuring web service datastores, see the *Designer Guide*.

7.4.4.2 To configure SSL in the runtime execution file

1. Obtain a certification authority (CA) certificate for the client.
2. Open `LINK_DIR\ext\webservice-c\axis2.xml` in a text editor.
3. Locate the commented `transportReciever` and `transportSender` elements in the XML:

```
<transportReceiver name="https" class="axis2_http_receiver">
  <parameter name="port" locked="false">6060</parameter>
  <parameter name="exposeHeaders" locked="true">false</parameter>
</transportReceiver>

<transportSender name="https" class="axis2_http_sender">
  <parameter name="PROTOCOL" locked="false">HTTP/1.1</parameter>
</transportSender>
```

4. Remove the comment (`<!-- -->`) tags around the `transportReciever` and `transportSender` elements.
5. Provide the path to the CA certificate as the `SERVER_CERT` parameter.

```
<parameter name="SERVER_CERT"></path/to/ca/certificate></parameter>
```

6. If you need client authentication, additionally provide the private key and keystore passphrase.

```
<parameter name="KEY_FILE"></path/to/client/certificate/chain/file></parameter>
<parameter name="SSL_PASSPHRASE"><passphrase></parameter>
```

7.4.5 Enabling WS-Security support

WS-Security is a communications protocol that applies security to web services at the message level. The protocol defines how integrity and confidentiality can be enforced on web service messaging, as well as how to attach signatures and encryption headers to SOAP messages. In addition, it defines how to attach security tokens such as X.509 certificates or Kerberos tickets to messages.

SAP Data Services implements WS-Security support through the Apache open source project rampart/c and can be configured through the web service datastore and an external policy file.

7.4.5.1 To configure WS-Security on the native web service datastore

1. Create a security policy file (`policy.xml`) based on the WS-Security policy specification that satisfies your security requirements.

   You can see sample policy files by downloading a copy of rampart/c and opening the `samples/secpolicy` folder.
2. Place your security policy file in the `LINK_DIR/ext/webservice-c` folder or another location.
3. If your policy file is not named `policy.xml` or is located in a folder other than the default location, specify the path using the WSS Policy file path parameter in the datastore configuration.

4. Enable the rampart/c module in your `axis2.xml` file.

   Within `axis2.xml`, uncomment the `<!-- module ref="rampart" /-->` line. By default, `axis2.xml` is installed to `LINK_DIR/ext/webservice-c/`.

You can also make more changes in the Advanced section of the native web service datastore configuration:

Parameter	Details
WSS Username	Enter the username to use for WS-Security.
WSS Password	Enter the password to use for WS-Security.
WSS Password type	Enter the password type to use for WS-Security. The available options are **`PlainText`** and **`Digest`**.
WSS Time to live	Enter the time for WS-Security protected messages to live. The default is 0. Any positive number will add a timestamp to the message.
WSS Policy file path	Enter the path to your WS-Security policy file. The default path is `LINK_DIR/ext/webservice-c/policy.xml`.

For more information about configuring web service datastores, see the *Designer Guide*.

Related Information

WS-Security policy specification

7.5 Using the operational statistics API

Use the operational statistics as web services API to retrieve job execution data from the Data Services repository and local log files.

You can use the following operations to generate dashboards to further analyze and review operational statistics:

Operation	Description
Get_Repository_List	Use the Get_Repository_List operation to get the entire repository list based on the user's permissions. This operation also returns a list of the repositories that the user is currently logged into. This operation is part of the Repo_Operations port.
Get_BatchJob_ExeDetail	Retrieves a list of all job executions for the selected repository and job. You can also filter the list by time range. This operation is part of the Batch_Job_Admin port.
Get_BatchJob_FlowDetails	Retrieves details about the tasks included in a job. This operation is part of the Batch_Job_Admin port.
Get_DF_Auditdata	Shows audit information for a data flow. This operation is part of the Batch_Job_Admin port.

Operation	Description
Get_DF_Monitor_Log	Shows the runtime statistics for single data flow execution. This operation is part of the Batch_Job_Admin port.
Get_DF_Monitor_Log_Pivot	Shows the runtime statistics as pivoted for a single data flow execution. This operation is part of the Batch_Job_Admin port.

Related Information

Get_Repository_List [page 2253]
Get_BatchJob_ExeDetail [page 2229]
Get_BatchJob_FlowDetails [page 2231]
Get_DF_Auditdata [page 2235]
Get_DF_Monitor_Log [page 2237]
Get_DF_Monitor_Log_Pivot [page 2239]

7.5.1 Prerequisites

The operations you use to retrieve operational statistics data are session security enabled.

Before you can call the functions, you need to log onto the system by using the Logon operation.

To disable session security, do the following:

1. Open Data Services Management Console.
2. Open the *Administrator* and then click *Web Services*.
3. Click the *Web Services Configuration* tab and select the operations for which you want to disable session security.
4. Choose the *Disable Session Security* option from the drop-down box at the bottom of the page and click *Apply*.

Endpoints

The WSDL URL depends on your web server host and port. You can obtain the endpoint URL from the Data Services Management Console.

Note

WSDL file version 2.1 supports the operations used to retrieve operational statistics data. Earlier versions do not support these operations.

The basic format for the URL is as follows:

```
http://<serverhost>:<port>/DataServices/servlet/webservices?ver=2.1
```

For example:

`http://localhost:8080/DataServices/servlet/webservices?ver=2.1`

To get the WSDL file, you need to add "&wsdlxml" to your endpoint. For example:

`http://localhost:8080/DataServices/servlet/webservices?ver=2.1&wsdlxml`

Related Information

Logon [page 2226]

7.6 Using the Message Client API

You can integrate SAP Data Services' real-time services by using the C++ or Java API. Either of these interfaces allows you to connect to the real-time service with a persistent connection to the server, send and receive data from it, and close the connection.

> i Note
>
> The Message Client API supports the creation of reports, similar to any job you run with the software.

In the execution of real-time jobs with real-time services APIs, these steps take place:

1. An administrator logs into the Management Console and chooses which real-time jobs to expose as real-time services. Those job names are stored in the local repository.
2. An administrator chooses which Access Server to run the services on and starts the real-time services.
3. A developer accesses a real-time service through Java and C++ libraries.
4. A C++ or Java application client makes a connection to the Access Server, which then sends information to various job servers.
5. The job servers route requests to an engine to process the real-time job.

File location

The Message Client API files for each supported platform are installed to **<LINK_DIR>**`\SDK\RTSDK`. When the software is installed on a Windows server, the Message Client API files for both C++ and Java for each UNIX platform are provided in a `.tar.gz` archive.

To use the UNIX Message Client API files with a Windows installation, copy the appropriate Message Client API package file for your UNIX platform from `LINK_DIR\SDK\RTSDK\platform` to your UNIX system, and then unzip and extract the archive to the desired installation location. For example, on Solaris:

```
gunzip MessageClient_Solaris_64bit.tar.gz
tar -xvf MessageClient_Solaris_64bit.tar
```

7.6.1 Interface components

The interface between the Access Server and your application includes these components:

- Connection definition (Connection)
  A class that defines the connection that your application uses to send and receive messages from the Access Server. Initialize the class (using the connect method) each time you initialize your application.
- Connection initialization (Connect)
  A method that creates the connection using host and port information supplied by the client.
- Request (Invoke)
  A method that indicates the request message for the specified real-time service. This method is a synchronous call that waits for a return.
- Exception handlers (Error message)
  A class that returns exceptions thrown by the connection object and system exceptions, if available.

7.6.2 Creating the connection

The Connection object creates an active connection to the Access Server.

Creating a Connection (calling the Connect method) does the following:

- Authenticates the client as secure
- Produces an open TCP/IP socket between the client and the Access Server
- Encapsulates the connection information into a client identifier (Connection ID)

As soon as you create the Connection object, you can use it to send messages to the Access Server. Typically, you would create a single Connection per client. If you attempt to call the Connect method for a Connection that already exists, the Access Server ignores the call.

7.6.3 Sending messages

Send requests from the client application using the Invoke method and the Connection ID.

Each business operation implemented by your web application can result in a call to the Access Server with a message. The Access Server uses the name of the business operation to determine the path for the message. When you use SAP Data Services to process real time jobs, you pair this business operation name, called a service, with the job and data flow names you defined in the software to process the message. There is a one-to-one correlation between business operation, service, job, and XML source.

Call the Invoke method with a string return value to process a synchronous response.

7.6.4 Closing the connection

The library provides a method (Disconnect) with the Connection object that allows you to systematically close the TCP/IP socket between the client and the Access Server.

7.6.5 Pseudo code example

```
// Login and authenticate the client connection =
connect(     accessServerAddress,

// TCP/HTTP address
clientName,
// matches Access Server
clientPassword);
// IP & Client
// security settings
// Invoke Service String xmlOut =
connection.invoke(     serviceName,

// has mapping to RT job     xmlIn);
// according to the RT job DTD
// In case of an error returns the error code
// and error message
```

7.6.6 C++ API reference

7.6.6.1 Class RTServiceClient

RTServiceClient

Contains C++ methods for allowing a client to connect to real-time services.

Method summary	
`virtual void`	connect (char* **`<hostname>`** unsigned short **`<port>`** bool **`<use_SSL>`** char* **`<trusted_certs_filename>`**)
`virtual char*`	invoke (char* serviceName char* inData)
`virtual void`	disconnect ( )

Constructor detail

RTServiceClient

```
RTServiceClient(){}
```

Method detail

connect

```
virtual void    connect(char* hostname, unsigned short port)
```

Establishes a connection between a client and the Access Server. You must establish a connection before a message can be exchanged.

Option	Description
hostname	The name or IP address of the machine that hosts the Access server.
port	The port number used for the connection.

setUseSSL

```
void setUseSSL(const char* certificate_dir)
```

Uses SSL when communicating with the Access Server.

`certificate_dir` – the folder that contains the certificate files. Files with `.crt`, `.pem`, and `.cer` extensions are used as trusted certificates.

invoke

```
virtual char* invoke(char* serviceName, char* inData)
```

Sends the input data to the real-time service and returns the output data.

Option	Description
serviceName	The name of the real-time service to invoke.
inData	The input data to send to the real-time service.

disconnect

```
virtual void disconnect ()
```

Stops the connection between a client and the Access Server.

7.6.6.2 Class RTServiceClientError

RTServiceClientError

Represents an error object thrown by the C++ class RTServiceClient.

Method summary
RTServiceClientError(const char*, int=0)
RTServiceClientError(const char*, const char*, const char*)

Method summary
RTServiceClientError(const RTServiceClientError&)

Method detail

RTServiceClientError

```
RTServiceClientError(const char*, int=0)
```

```
RTServiceClientError(const char*, const char*, const char*)
```

```
RTServiceClientError(const RTServiceClientError&)
```

Represents an error object thrown by the client library.

7.6.7 Java API reference

7.6.7.1 Class RTServiceClient

```
com.businessobjects.rtsclient.RTServiceClient
```

Contains Java methods for allowing a client a connection to real-time services.

Method Summary	
`public void`	connect (char * **`<machineName>`** int **`<port>`** bool **`<use_SSL>`** char* **`<trusted_certs_filename>`**) throws RTServiceExceptionThrows:
`public java.lang.String`	invoke (java.lang.String **`<serviceName>`** java.lang.String **`<inData>`**) throws RTServiceExceptionThrows:
`public void`	disconnect ( ) throws RTServiceExceptionThrows:

Method detail

connect

```
public void    connect (java.lang.String machineName, int port, bool use_SSL, char*
trusted_certs_filename) throws RTServiceExceptionThrows:
```

Establishes a connection between a client and the Access Server. You must establish a connection before a message can be exchanged.

Option	Description
machineName	The name or IP address of the machine that hosts the Access Server
port	The port number used for the connection.
use_SSL	Indicates that SSL should be used for the connection to the Access Server.
trusted_certs_filename	The path to the file that contains the trusted SSL certificates.

invoke

```
public java.lang.String invoke(java.lang.String serviceName, java.lang.String
inData) throws RTServiceExceptionThrows:
```

Sends the input data to the real-time service and returns the output data.

Option	Description
serviceName	The name of the real time service to invoke.
inData	The input data to send to the real time service.

disconnect

```
public void disconnect ()throws RTServiceExceptionThrows:
```

Stops the connection between a client and SAP Data Services.

7.7 Using the JMS adapter

7.7.1 Introduction

7.7.1.1 About this section

This section provides a detailed step-by-step method of installing and configuring the SAP Data Services JMS adapter. It includes a description of required support software, including supported versions, details of the adapter components, environment setup both for the software and external applications, and instructions for executing the adapter.

7.7.1.1.1 Who should read this section?

This section assumes the following:

- You understand how to use Designer to design and run real-time data flows (RTDFs) and batch jobs.
- You have a basic understanding of how to use Administrator to administer SAP Data Services processes. (You administer adapters from the Administrator.)

- You have a working knowledge of the environment this adapter is targeting.
- You know the role an adapter plays in business systems integration.
- You have some familiarity with XML and XML configuration schemas.
- Also, to integrate the software with an external system, it's recommended that you be familiar with systems administration and systems integration issues.

General SAP Data Services product documentation assumes the following:

- You are an application developer, consultant or database administrator working on data extraction, data warehousing, or data integration.
- You understand your source and target data systems, DBMS, legacy systems, business intelligence, and messaging concepts.
- You understand your organization's data needs.
- If you are interested in using this product to design real-time processing you are familiar with:
    - DTD and XML Schema formats for XML files
    - Publishing Web Services (WSDL, HTTP/S and SOAP protocols, etc.)
- You are familiar with the software's installation environments: Microsoft Windows or UNIX.

7.7.1.2 Adapter overview

Typical enterprise infrastructure is a complex mix of off-the-shelf and custom applications, databases, ERP applications etc. SAP Data Services combines and extends critical Extraction Transformation Loading (ETL) and Enterprise Application Integration (EAI) technology components required for true enterprise data integration.

Integrating disparate applications with the software's platform requires adapters. These adapters help facilitate otherwise incompatible applications and systems work together, thereby sharing data.

7.7.1.2.1 About Java Messaging Service (JMS)

Enterprise-messaging or Message Oriented Middleware (MOM) products are fast becoming an essential component for integrating intra-company operations. They allow separate business components to be combined into a reliable, yet flexible, system. In addition to the traditional MOM vendors, several database vendors and Internet-related companies also provide enterprise-messaging products.

Java language clients and Java language middle-tier services must be capable of using these messaging systems. Java Messaging Service (JMS) provides a common way for Java language programs to access these systems.

JMS is a set of interfaces and associated semantics that define how a JMS client accesses the facilities of an enterprise-messaging product. Since messaging is peer-to-peer, all users of JMS are generically referred to as clients. A JMS application is composed of a set of application-defined messages and a set of clients that exchange them. Products that implement JMS do this by supplying a provider that implements the JMS interfaces.

7.7.1.2.2 Scope of the JMS adapter

- SAP Data Services initiates Request/Reply
  The software initiates the request by sending the message on a pre-configured request queue and gets the reply on a pre-configured reply queue.
- The software initiates Request/Acknowledgment
  The software initiates the request by sending the message on a pre-configured target queue or by publishing a message to a JMS topic. In this case, only the acknowledgment is sent back to the software.
- IR initiates Request/Acknowledgment & Request/Reply
  In this case, an external Information Resource (IR is a JMS compatible application) sends the requests to the software and gets the reply or acknowledgment.
  Alternatively, the IR publishes a message to a JMS topic to which the JMS adapter has subscribed.

7.7.2 Installation and configuration

7.7.2.1 JMS adapter installation

This section details the components of the Adapter for JMS as well as system requirements.

The Adapter for JMS is automatically installed when you install SAP Data Services version 12.0.0 or later.

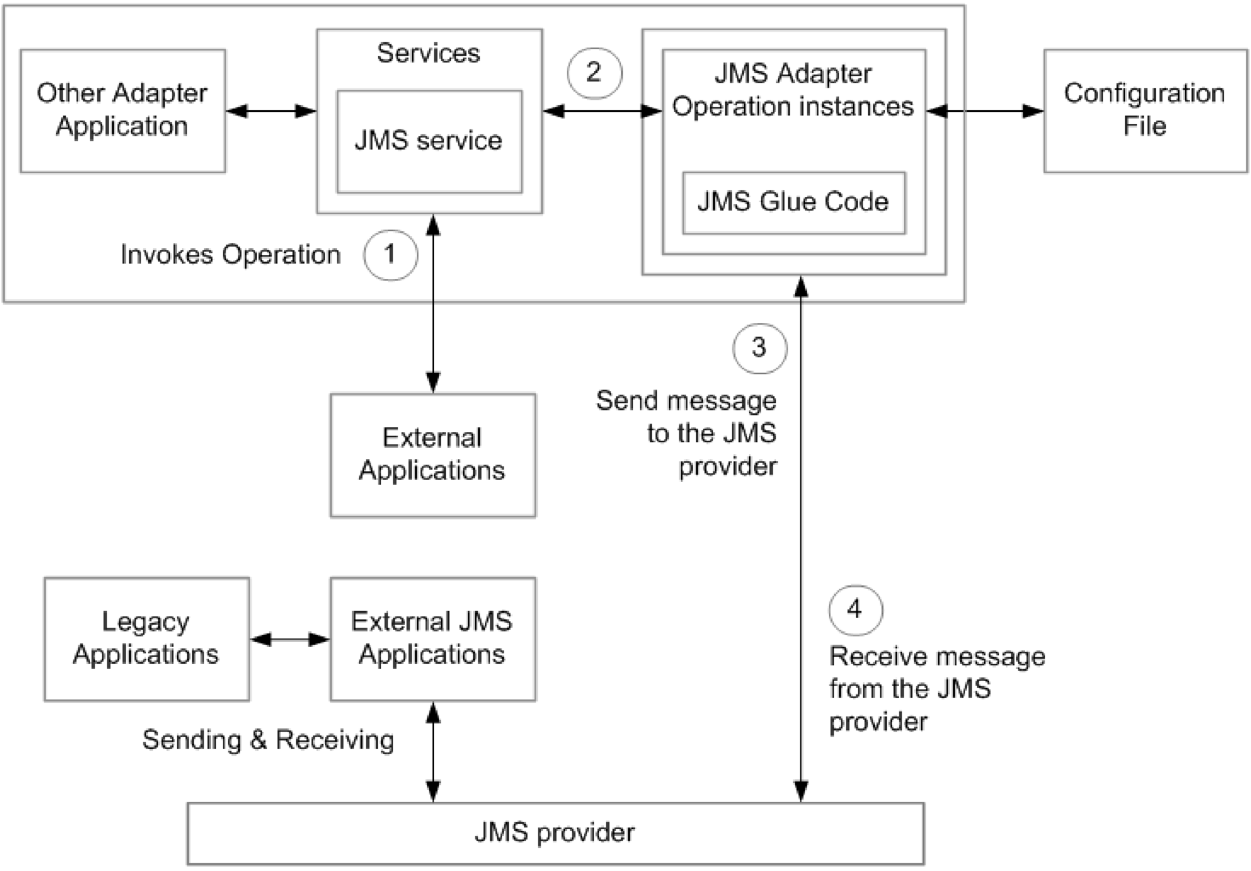
7.7.2.1.1 System prerequisites

Before you install your SAP Data Services Adapter for JMS, ensure that the following software is installed. For specific installation instructions, see the technical documentation for each product.

Software	Version	Comment
JMS Provider		For example, Weblogic Application Server
SAP Data Services	11.7.0 or later	Use the software to configure the services and adapter
SAP Data Services Adapter SDK	2.0.0.0 or later	

7.7.2.1.2 Adapter product components

The following diagram shows a functional overview of the SAP Data Services Adapter for JMS with other components and their potential interrelationships:


Other Adapter Application
Services
JMS service
2
JMS Adapter Operation instances
JMS Glue Code
Configuration File
Invokes Operation
1
3
Send message to the JMS provider
External Applications
Legacy Applications
External JMS Applications
4
Receive message from the JMS provider
Sending & Receiving
JMS provider

The diagram shows the architecture and functionality of the SAP Data Services Adapter for JMS as well as how the adapter interacts with the external JMS application through the JMS Provider. The adapter sends or receives data on queues using the Point to Point (P2P) mode of communication, or publishes or subscribes to a JMS topic using the Publish/Subscribe mode of communication.

The flow of control in the previous diagram is as follows:

1. External application invokes a service on the software.
2. Based on the service invoked on the software, its respective real-time data flow (RTDF) invokes the Operation instance with XML data sent by the external application as input.
3. This operation instance sends the message to the configured queue or topic in the JMS provider. Based on the type of operation (such as Request/Reply or Request/Acknowledge), the JMS provider sends the Reply/Acknowledgment message back to the software.
4. External JMS application sends messages to the JMS Provider on a request queue or publishes the message to a topic. The JMS Adapter receives these messages after polling them from the JMS Provider and for P2P, sends the reply back to external JMS application on a configured reply queue. No reply is sent if the message was from a topic.

7.7.2.2 JMS adapter configuration

Before the Adapter for JMS can begin integrating the JMS Provider with the SAP Data Services system you must create and configure at least one adapter instance and at least one operation for each instance. Adapter instances identify the JMS Application used in the integration. Adapter operations identify the integration operations to be used with the configured adapter instance.

Operations provided with Adapter for JMS include the following:

- PutGet Operation (Request/Reply): The software initiates a request, sending a message on a pre-configured request queue. Simultaneously, the software listens on a pre-configured reply queue. An external JMS-compatible application listens on the request queue, processes the request, and returns an XML response message to the reply queue. The adapter sends the message to the Job service.
- Put Operation (Request/Acknowledgment): The software initiates a request, sending a message on a pre-configured target queue. If the message was sent successfully, the adapter sends an acknowledgement to the Job service. The adapter raises an exception if it was unable to send the message.
- Get Operation (Request/Acknowledgment and Request/Reply from Information Resource): An external information resource (IR) sends a request XML message to a JMS queue. The adapter polls the JMS queue at a time interval you specify in the configuration. When the adapter receives a message from the JMS queue, it sends the message to the pre-configured Job service.
  After processing the XML message, the Job service may send a response message to the adapter. When this happens, the adapter puts the message in a pre-configured response queue. If the response queue is not configured, it becomes a request/acknowledgment operation and no reply is sent. If there is any error in invoking another service from the Job service, the original message is sent to the undelivered queue for reference by the IR.
- PutTopic Operation (Request/Acknowledgment): A software service initiates a request, publishing an XML message to a pre-configured target topic. If the message was sent successfully, the adapter sends an acknowledgement to the Job service. The adapter raises an exception if it was unable to send the message.
- GetTopic Operation (Request/Acknowledgment): An external information resource (IR) publishes an XML message to a JMS topic. The adapter polls the topic at the time intervals specified in the configuration. When the adapter receives the message from the topic, it sends the message to the service that handles the message.

7.7.2.2.1 To configure the JMS adapter

All SAP Data Services adapters communicate with the software through a designated Adapter Manager Job Server. Install adapters on the computer containing your designated Adapter Manager Job Server. This special Job Server integrates adapters with the software using the Administrator and Designer. After you install your adapter:

1. Use the Server Manager utility to configure adapter connections with the Adapter Manager Job Server.
2. From the Administrator, perform the following tasks:
   - Add at least one instance of the adapter to system.
   - Add at least one operation for each adapter instance.
   - Start the adapter instance (operations are started automatically).
3. Open the Designer and create an adapter datastore. Use metadata accessed through the adapter to create batch and/or real-time jobs.

For more information, see “To configure Job Servers” in the *Installation Guide* and “Adapter Considerations” in the *Management Console Guide*.

7.7.2.2.2 To configure an adapter instance in the Administrator

From the Administrator you can add a JMS adapter to the SAP Data Services system as well as edit existing adapter configurations. Add the adapter in the Administrator before you run jobs that use information from that adapter.

1. Select *Adapter Instances* › *Job Server*.
2. Select the *Configuration* tab.
3. Click *Add*.
4. Select *JMSAdapter* from the list of adapters available on this Job Server and click *Apply*.
5. Enter the required information to create a JMS Adapter instance and click *Apply*.

The Administrator makes the adapter instance available to the software.

7.7.2.2.2.1 Adapter instance configuration information

To configure a JMS adapter instance in SAP Data Services, you need to complete the fields in the Administrator under Adapter instance startup configuration.

Field	Description
Adapter Instance Name	Enter a unique name that identifies this instance of the adapter.
Access Server Host	Enter the host ID of the computer running the Access Server that connects to this adapter instance. To run a real-time job, you must configure a service that the Access Server will use to run the job. When a job uses adapter-based data, the Access Server must be able to connect to the adapter instance.
Access Server Port	The message broker port of the Access Server host. After you log into the Administrator for this Access Server, select *Configuration* › *Client Interfaces* to view message broker port information.
Adapter Retry Count	Applies if adapter instance fails or crashes. Enter 0 for no retries; enter a negative number for indefinite retries.
Adapter Retry Interval	Wait in msec. between adapter retry attempts.
Classpath	The adapter is a Java program, so you must configure the jar files required by the adapter `CLASSPATH`. The adapter is pre-configured with most of the necessary jar files, except for the vendor-specific JMS provider jar files and the `j2ee.jar` file. Add these jar files to the `CLASSPATH`. For example: • **`<LINK_DIR>`**`/lib/acta_adapter_sdk.jar` • **`<LINK_DIR>`**`/lib/acta_broker_client.jar` • **`<LINK_DIR>`**`/lib/acta_tool.jar`

Field	Description
	• <LINK_DIR>/ext/lib/xerces.jar • <LINK_DIR>/lib/acta_jms_adapter.jar • <LINK_DIR>/ext/lib/jms/<JMS Provider Jar File> • <LINK_DIR>/ext/lib/jms/j2ee.jar Note Specify the jar file provided with the JMS provider that you are using. For Weblogic, the name of jar file is weblogic.jar. Note The j2ee.jar file is required. You can get the j2ee.jar file from Java EE 1.6 and copy it to the adapter job server machine. You then need to add the j2ee.jar to the JMS adapter CLASSPATH.
Autostart	When set to True, the adapter interface automatically starts when the Administrator starts.
Trace Mode	Set this flag to control the number of trace messages the adapter produces. There are two settings: • True: The adapter interface writes information and error messages to help debug problems. The adapter writes information and error messages to the <adapter_instance_name>_trace.txt file in the <DS_COMMON_DIR>\adapters\logs directory. • False: The adapter interface writes only error information messages. The adapter writes error messages to the <adapter_instance_name>_error.txt file in the <DS_COMMON_DIR>\adapters\logs directory.
Additional Java Launcher Options	Additional command line parameters used for the javaw.exe command line and for the adapter itself. (See specific adapter documentation for details.)
Adapter Type Name	(Read-only) Name of the adapter used to create this instance.
Adapter Version	(Read-only) Version of the adapter used to create this instance.
Adapter Class	(Read-only) Name that identifies the adapter class. The name depends on the type of adapter.

In the JMS Adapter section, select a Configuration Type and enter Configuration parameters.

Parameter	Description
Configuration Type	Uses only the configuration parameters associated with the selected configuration type. ● JNDI configuration type ● MQ configuration type

For the JNDI configuration type, use the following configuration parameters.

Parameter	Description
Server URL	Represents the URL of the JMS Provider. For example: `t3://`**`<JMS Provider IP Address>`**`:`**`<port number>`**.
JNDI Context Factory	JNDI context factory name is JMS Provider specific. You can choose the context factory from a list that includes common context factories. If you require a context factory that is not listed, you can add it to the list by editing file **`<DS_COMMON_DIR>`**`/adapters/config/templates/JMSAdapter.xml` and updating the `<jndiFactory>` element. For Weblogic as a JMS Provider, the JNDI Factory name is: `weblogic.jndi.WLInitialContextFactory`.
Queue Connection Factory	Queue connection factory name. For example: `JMSConnections.AdapterConnectionFactory`.
Topic Connection Factory	Topic connection factory name. For example: `JMSConnections.AdapterTopicConnectionFactory`.

For the MQ configuration type, use the following configuration parameters.

Parameter	Description
MQ Queue Manager Name	(Optional) Specify if not using the default MQ Queue Manager on the system running MQ.
MQ Channel Name	(Optional) Specify if not using the default MQ Channel on the system running the adapter.
MQ Computer Name	(Optional) Specify if not using the MQ Queue Manager on the same system running the adapter.
MQ Port	(Optional) Specify if not using the default MQ port (1414).
MQ User ID	(Optional) Specify if required to login to the MQ Queue Manager.
MQ Password	(Optional) Specify if required to login to the MQ Queue Manager.

7.7.2.2.3 To add an operation instance to an adapter instance

1. Select *Configuration* > *Adapter instances* .
2. Click *Operations* under Dependent Objects.
3. Click *Add* to configure a new operation. Or, you can click the link of an existing operation instance to edit its configuration.
4. Select an operation type from the list and click *Apply*. The options that appear on this page vary based on operation-specific design.

Complete the operation instance configuration form and click *Apply*.

7.7.2.2.3.1 Operation instance configuration options

Each operation type contains different configuration options. Operations include:

- Put Operation (request/acknowledgment) options
- PutTopic Operation (request/acknowledgment) options
- PutGet Operation (request/reply) options
- Get Operation (request/reply and request/acknowledgment) options
- GetTopic Operation (request/acknowledgment only) options

Note

When specifying a queue or topic, you must provide the JNDI queue name or the MQ queue name as indicated by the Adapter Configuration Type property.

7.7.2.2.3.1.1 Put Operation (request/acknowledgement) options

To set up an operation instance of type Put Operation in SAP Data Services, complete the following fields in the Administrator.

Field	Description
Operation instance	The unique operation instance name. In the Designer, your operation metadata object is imported with this name.
Thread count	The number of copies of Request/Reply operation to run in parallel. For parallel (asynchronous) processing of messages coming from real-time service, more than one copy should be used. But if the sequence of messages is important (synchronous processing), more than one thread should not be used. (Multiple copies of real-time services must be

FIeld	Description
	supported by multiple instances of Request/Reply.) The default is 1.
Operation retry count	The number of times to retry this operation if it fails. Enter 0 to indicate no retries are to be attempted. Enter a negative number to indicate the operation should be retried indefinitely.
Operation retry interval	The time (in milliseconds) to wait between operation retry attempts.
Display name	The display name of the operation instance. This display name is visible in the Designer's metadata browsing window.
Description	The description of the operation instance. This description is visible in the Designer's metadata browsing window.
Enable	Whether to enable the operation to start at the same time as the adapter instance. Valid values are true and false. ● When true, the operation starts when the adapter instance starts. ● When false, the operation needs to be started manually from *Adapter Operations Status* window of the adapter administrator.
Destination Queue	The name of the destination queue where the message will be sent.
Request Format	The DTD or XSD file name that defines the XML message used in the operation.
Request XML Root Element	The name of the XML root element.

7.7.2.2.3.1.2 PutTopic Operation (request/ acknowledgement) options

To set up an operation instance of type PutTopic in the SAP Data Services, complete the following fields in the Administrator.

Field	Description
Operation instance	The unique operation instance name. In the Designer, your operation metadata object is imported with this name.
Thread count	The number of copies of Request/Reply operation to run in parallel. For parallel (asynchronous) processing of messages coming from real-time service, more than one copy should be used. But if the sequence of messages is important (synchronous processing), more than one thread should not be used. (Multiple copies of real-time services must be supported by multiple instances of Request/Reply.) The default is 1.

Field	Description
Operation retry count	The number of times to retry this operation if it fails. Enter 0 to indicate no retries are to be attempted. Enter a negative number to indicate the operation should be retried indefinitely.
Operation retry interval	The time (in milliseconds) to wait between operation retry attempts.
Display name	The display name of the operation instance. This display name is visible in the Designer's metadata browsing window.
Description	The description of the operation instance. This description is visible in the Designer's metadata browsing window.
Enable	Whether to enable the operation to start at the same time as the adapter instance. Valid values are true and false. ● When true, the operation starts when the adapter instance starts. ● When false, the operation needs to be started manually from *Adapter Operations Status* window of the adapter administrator.
Destination Topic	The topic to which the operation is published. Use JNDI or MQ name as specified by Adapter Configuration Type.
Message Format	The DTD or XSD file name defining the XML message used in this operation.
Request XML Root Element	The name of the XML root element.
Persistent Message	Whether to make published messages available to durable subscribers. Valid values are true and false. When true, published messages are available to durable subscribers.

7.7.2.2.3.1.3 PutGet Operation (request/reply) options

To set up an operation instance of type PutGet Operation in SAP Data Services, complete the following fields in the Administrator.

Field	Description
Operation instance	The unique operation instance name. In the Designer, your operation metadata object is imported with this name.
Thread count	The number of copies of Request/Reply operation to run in parallel. For parallel (asynchronous) processing of messages coming from real-time service, more than one copy is used. If the sequence of messages is important (synchronous processing), more than one thread should not be used. (Multiple copies of real-time services must be supported by multiple instances of Request/Reply.) The default is 1.
Operation retry count	The number of times to retry this operation if it fails. Enter 0 to indicate no retries are to be attempted. Enter a negative

Field	Description
	number to indicate the operation should be retried indefinitely.
Operation retry interval	The amount of time (in milliseconds) to wait between operation retry attempts.
Display name	The display name of the operation instance. This display name is visible in the Designer's metadata browsing window.
Description	The description of the operation instance. This description is visible in the Designer's metadata browsing window.
Enable	Whether to enable the operation to start at the same time as the adapter instance. Valid values are true and false. ● When true, the operation starts when the adapter instance starts. ● When false, the operation needs to be started manually from *Adapter Operations Status* window of the adapter administrator.
Request Queue	The name of the destination queue where the message will be sent.
Reply Queue	The name of the destination queue where the message will be sent.
Timeout	The maximum time (in milliseconds) the operation should wait for the reply message.
Continue After Error	Whether to continue after encountering an error. Valid values are true and false. ● When true, the operation instance remains in start stage even after the error. ● When false, the operation instance stops after the error occurs during the process.
Request Format	The DTD or XSD file name that defines the Request XML message used in this operation.
Request XML Root Element	The name of the XML root element in the Request DTD or XSD.
Reply Format	The DTD or XSD file name that defines the Reply XML message used in the operation.
Reply XML Root Element	The name of the XML root element in the Reply DTD or XSD.

7.7.2.2.3.1.4 Get Operation (request/reply and request/acknowledgement) options

To set up an operation instance of type Get Operation in SAP Data Services, complete the following fields in the Administrator.

Field	Description
Operation instance	The unique operation instance name. In the Designer, your operation metadata object is imported with this name.
Polling interval	The time interval (in milliseconds) for polling the source queue by this operation instance. For example, If the polling interval is 1000, then it polls the source queue after every one second.
Operation retry count	The number of times to retry this operation if it fails. Enter 0 to indicate no retries are to be attempted. Enter a negative number to indicate the operation should be retried indefinitely.
Operation retry interval	The time (in milliseconds) to wait between operation retry attempts.
Enable	Whether to enable the operation to start at the same time as the adapter instance. Valid values are true and false. ● When true, the operation starts when the adapter instance starts. ● When false, the operation needs to be started manually from *Adapter Operations Status* window of the adapter administrator.
Source Queue	The name of the queue where the message is sent by the IR and received by the adapter. Use JNDI or MQ name as specified by the Adapter Configuration Type.
Service	The name of the real-time service invoked by the operation when it receives a new message from the Source Queue.
Timeout	The maximum time (in milliseconds) that the Service takes to process a message. If the operation instance is unable to invoke the service within the Timeout limit, it sends the error message to the undelivered queue.
Continue After Error	Whether to continue after encountering an error. Valid values are true and false. ● When true, the operation instance remains in start stage even after the error. ● When false, the operation instance stops after the error occurs during the process.
Default Response Queue	[optional]: Used only for Request/Reply operation. In case of Request/Acknowledgment operation, it remains blank. The application sends the reply back to external JMS application (IR) on this queue. Use JNDI or MQ name as specified by the Adapter Configuration Type.
Undelivered Queue	[optional]: The undelivered queue for receiving the error messages, if any. Use JNDI or MQ name as specified by the Adapter Configuration Type.
Request DTD Root Element	The name of the root element for the input DTD for this operation.

7.7.2.2.3.1.5 GetTopic Operation (request/ acknowledgement only) options

To set up an operation instance of type GetTopic in SAP Data Services, complete the following fields in the Administrator.

Field	Description
Operation instance	The unique operation instance name. In the Designer, your operation metadata object is imported with this name.
Polling interval	The time interval (in milliseconds) for polling the source topic by this operation instance. For example, if the polling interval is 1000, then it polls the source topic after every one second.
Operation retry count	The number of times to retry this operation if it fails. Enter 0 to indicate no retries are to be attempted. Enter a negative number to indicate the operation should be retried indefinitely.
Operation retry interval	The time (in milliseconds) to wait between operation retry attempts.
Enable	Whether to enable the operation to start at the same time as the adapter instance. Valid values are true and false. ● When true, the operation starts when the adapter instance starts. ● When false, the operation needs to be started manually from *Adapter Operations Status* window of the adapter administrator.
Source Topic	The topic to which the operation subscribes. Use JNDI or MQ name as specified by Adapter Configuration Type.
Durable subscriber	The subscription name of Durable subscriber. If not applicable, leave this field blank.
Service	The name of the real-time service invoked by the operation when it receives a new message from the source topic.
Timeout	The maximum time (in milliseconds) that the service takes to process a message.
Continue After Error	Whether to continue after encountering an error. Valid values are true and false. ● When true, the operation instance remains in start stage even after the error. ● When false, the operation instance stops after the error occurs during the process.

7.7.2.2.4 Defining a JMS adapter datastore

Use the SAP Data Services Adapter for JMS with a batch job or real-time data flow (RTDF) when the batch job or RTDF passes a message to an operation instance, using either:

- An Outbound message (for Request/Acknowledge operations)
- A Message Function (for Request/Reply operations)

You must first define an adapter datastore in the Designer. Then, the batch job or RTDF can pass a message to one of the adapter operation instances defined in that datastore. To define an adapter, you must:

- Define a datastore object for each adapter instance
- Define one function or one outbound message for each operation instance to which you want to pass a message.

For each adapter instance, define a corresponding datastore object in the Datastore Editor window of the Designer object library.

7.7.2.2.4.1 To define a JMS adapter datastore

1. From the Datastore Editor:
   a) Select the *Job Server* configured to manage your JMS adapter.
   b) Select the *Adapter instance name* you configured in the Administrator.
2. Select the Adapter Properties tab and enter values for each property.
3. Click *OK* to save values and create the datastore.

7.7.2.2.5 Importing message functions and outbound messages to the datastore

You can pass messages from a batch job or RTDF to an operation instance. Import either a function or an outbound message (depends on the type of operation involved) in the Designer Datastore library for each operation instance.

Real-time data flows use following methods.

Method	Description
Message functions	Pass messages to an operation instance if the RTDF waits for a return XML message from the IR.
Outbound messages	Outbound messages Pass messages to an operation instance if the RTDF waits for a confirmation only (not a return XML message) from the IR.

Operation types in the SAP Data Services Adapter for JMS have the following invocation types.

Operation type	Invocation type
Request/Reply Operation	Message Function
Request/Acknowledge Operation	Outbound Message

7.7.2.2.5.1 To import message functions and outbound messages

1. In Designer, double-click the datastore associated with your JMS Adapter Instance to display the Adapter metadata browser window.
2. Right-click the operation instance to be imported and select Import.
   The selected operation instance is added to the datastore.

These message functions and outbound message functions can be used for creating Batch Jobs or RTDFs in SAP Data Services.

7.7.3 Using the JMS adapter

7.7.3.1 To start an instance of the JMS adapter

1. From the Administrator go to *Adapter Instance* > *Job Server* and select the *Status* tab.
2. Select the check-box next to the previously configured adapter instance.
3. Click *Start*.

When the adapter instance and its operations start, the message “Started” appears in the Status column.

7.7.3.1.1 Operations from SAP Data Services to the JMS adapter

7.7.3.1.1.1 Request/Reply - PutGet operation

SAP Data Services initiates the request by sending a message on a pre-configured request queue. Simultaneously, the software also listens on a pre-configured reply queue. An external JMS-compatible application listening on this request queue, after processing, sends back the response on response queue. This response, in the form of the reply XML message, is returned back to the software.

Related Information

Testing PutGet: Request/Reply [page 2290]

7.7.3.1.1.2 Request/Acknowledge - Put operation

SAP Data Services initiates the request by sending the message on a pre-configured target queue.

Related Information

Testing Put: Request/Acknowledge [page 2296]

7.7.3.1.1.3 Request/Acknowledge - PutTopic operation

SAP Data Services initiates the request by publishing the message to a pre-configured target topic.

Related Information

Testing PutTopic: Request/Acknowledge [page 2291]

7.7.3.1.2 Operations from Information Resource (IR) to Data Services

7.7.3.1.2.1 Request/Reply - Get operation

IR initiates the request by putting a message in the source queue of the Get operation. The Get operation receives the message from the source queue during a polling cycle and sends the message to the configured Job service. The service sends a reply message to the Get operation, which then puts the message in the response queue. The IR then gets the message from the response queue.

Related Information

Testing Get: Request/Reply [page 2292]

7.7.3.1.2.2 Request/Acknowledge - Get operation

IR initiates the request by putting a message in the source queue of the Get operation. The Get operation receives the message from the source queue during a polling cycle and sends the message to the configured Job service.

Related Information

Testing Get: Request/Acknowledge [page 2294]

7.7.3.1.2.3 Request/Acknowledge - GetTopic operation

IR initiates the request by publishing a message to the source topic of the GetTopic operation. The GetTopic operation receives the message from the source queue during a polling cycle and sends the message to the configured Job service.

Related Information

Testing GetTopic: Request/Acknowledge [page 2294]

7.7.3.2 To run the sample

This section details the JMS adapter operations.

1. Import the `JMSAdapter.atl` file into the Designer. Find the `.atl` file in **`<DS_COMMON_DIR>`**`/adapters/jms/samples`. The imported project name is Acta_JMSAdapter_Sample.
2. Change the input and output XML files path for all the batch jobs depending on your location of your **`<DS_COMMON_DIR>`** environment variable.
3. Use the Administrator Real-Time Services Configuration tab to create the service Queue.TestService referencing job TestService_Job and Topic.TestService referencing job TestServiceTopic_Job.
4. Open Web Administrator and configure a JMS adapter. Define the operations detailed in the following tests.
5. Use the Designer to edit the JMSAdapter datastore and rename it to the name of the adapter you just created.

Before running the sample, create the following queues and topic using your JMS provider utilities:

- Queue.MyQueue
- Queue.ActaQueuePutGet
- Queue.ActaQueuePutGet1
- Queue.ActaQueueGet
- Queue.ActaReplyQueueGet
- Queue.ActaUndeliveredQueue
- Topic.MyTopic

> Note
>
> The `JMSAdapterTest.properties` file and the scripts to execute the samples are located in the **`<DS_COMMON_DIR>`**`/adapters/jms/samples` directory.

The `JMSAdapterTest.properties` file `TopicConnectionFactoryName` property value is **`Tcf`** and the `QueueConnectionFactoryName` property value is **`Qcf`**. You must edit this file and change the property values if the adapter was configured using different factory names.

The `JMSAdapterTest.properties` file `MessageSource` property refers to the file **`<DS_COMMON_DIR>`**`/adapters/jms/samples/xml/JMSSource.xml`. You must edit this file and change the property value if this is not where your `JMSSource.xml` file is located.

You must edit `setTestEnv.bat` on Windows or `setTestEnv.sh` on UNIX to set the JMS Provider jar files in the class path used by the sample test programs.

7.7.3.2.1 Configuring the JMS provider

Create a JMS Server, Connection Factory and configure JMS queues to run SAP Data Services Adapter for JMS. For testing the adapter, using sample applications, configure the following queues and topic:

- Queue.MyQueue
- Queue.ActaQueuePutGet
- Queue.ActaQueuePutGet1
- Queue.ActaQueueGet
- Queue.ActaReplyQueueGet
- Queue.ActaUndeliveredQueue
- Topic.MyTopic

Refer to the “Appendix” section for instructions on using Weblogic as the JMS Provider. Steps for JMS Provider may differ from the example provided in this section.

7.7.3.2.2 To use MQ instead of JNDI configuration

The properties file used by the samples, `JMSAdapterTest.properties`, is set up to use the JNDI configuration. You can edit this file to use MQ configuration parameters.

1. Open the `JMSAdapterTest.properties` file.
2. Set ConfigType = MQ.
3. Set any of the following properties as required by your system:
    - MqQueueManager
    - MqChannel
    - MqComputerName
    - MqPort
    - MqUserID
    - MqPassword
4. For the queue and topic names, use MQ names instead of the JNDI names for the following properties:
    - TopicGetName
    - TopicPutName
    - QueueSourceGetName
    - QueueResponseGetName
    - QueuePutName
    - QueueRequestPutGetName
    - QueueReplyPutGetName

7.7.3.3 Testing PutGet: Request/Reply

To configure the operation type PutGet (Request/Reply), enter the following information in the operation instance configuration page in Administrator.

Option	Value
Operation instance	JMSPutGetOperation
Thread count	1
Display name	JMSPutGetOperation
Description	This operation instance represents the PutGet Request/Reply operation. It sends the request message to the request queue and receives the reply message from the reply queue.
Enable	true
Request queue	Queue.ActaQueuePutGet
Reply queue	Queue.ActaQueuePutGet1
Timeout	200000
Continue after error	true
Request format	**<DS_COMMON_DIR>**`/adapters/JMS/samples/dtd/JMSPUTGET_SOURCE.dtd`
Request XML root element	source
Reply format	**<DS_COMMON_DIR>**`/adapters/JMS/samples/dtd/JMSPUTGET_RESPONSE1.dtd`
Reply XML root element	source

After entering this information, click *Apply* and restart the JMS Adapter instance. When the JMS Adapter starts running, the operation instance also starts running.

Testing on Windows

Open a command prompt window and change directory to **<DS_COMMON_DIR>**`\adapters\jms\samples`. Run the sample application (external IR) by running `sampleTest_PutGet.bat`. The application displays the message:

```
Ready to receive message from queue Queue.ActaQueuePutGet
```

Execute the batch Job JMSPutGetOperation_BatchJob from the Designer. This sends the message to the request queue.

The sample application (external IR) listens for a message to arrive at the request queue of the JMSPutGetOperation instance. When it receives the message, it prints a message to the command prompt window such as:

```
Message received:  <?xml version="1.0" encoding="UTF-8"?>  <!-- Data Services
generated XML -->  <!-- 2005-05-05.16:41:57(539,223)[1] -->  <source>  <age>18</
age>  <salary>200000000</salary>  <acno>2356376438743</acno>  </source>
```

The sample test program then sends a reply message to the reply queue configured for the JMSPutGetOperation instance. It echoes a message to the command prompt window such as:

```
Message sent:  <?xml version="1.0" encoding="UTF-8"?>  <source>
<age>ReplyFromJMSIR1</age>  <salary>ReplyFromJMSIR2</salary>
<acno>ReplyFromJMSIR3</acno>  </source>
```

After the adapter operation receives the reply from the reply queue, it sends the message to the job which then generates the output file `JMSSourceOutput_PutGet.xml` under the directory **<DS_COMMON_DIR>**`/adapters/JMS/samples/xml`. The contents of the file should be similar to the message sent from the sample test with the addition of a timestamp and error information.

Testing on UNIX

Run the sample application (external IR) by running `sampleTest_PutGet.sh` file from the command prompt.

Execute the batch Job JMSPutGetOperation_BatchJob from Designer. This sends the message at the request queue.

Sample application (external IR) listens for the message at the request queue of JMSPutGetOperation instance and sends the message to the reply queue configured for the JMSPutGetOperation instance. After receiving the reply from the reply queue an output file `JMSSourceOutput_PutGet.xml` is generated under the directory **<DS_COMMON_DIR>**`/adapters/JMS/samples/xml`.

7.7.3.4 Testing PutTopic: Request/Acknowledge

To configure the operation type Put topic (Request/Acknowledge), enter the following information in the operation instance configuration page in the Web Administrator.

Option	Value
Operation instance	JMSPutTopicOperation
Thread count	1
Operation retry count	5
Operation retry interval	15000
Display name	JMSPutTopicOperation Display Name
Description	JMSPutTopicOperation Display Name
Enable	true
Destination queue	Topic
Message format	`C:\ProgramFiles\SAP BusinessObjects\Data Services`
Request XML root element	source
Persistent message	true

After entering this information, click *Apply* and restart JMS Adapter instance. When the JMS Adapter starts running, the operation instance also starts running.

Add the testing sections:

Testing on Windows

Open a command prompt window and change directory to **<DS_COMMON_DIR>**\adapters\jms\samples. Run the sample application (external IR) by running sampleTest_PutTopic.bat. The application should display the message:

```
Ready to receive message from topic Topic.MyTopic
```

If you do not see this message, then start the JMS publish/subscribe broker. The message should appear after you start the broker.

Execute the batch Job JMSPutTopicOperation_BatchJob from the Designer.

The sample application (external IR) listens for a message to be published by the JMSPutTopicOperation instance. When it receives the message, it will print a message to the command prompt window such as:

```
Received message:  <?xml version="1.0" encoding="UTF-8"?>  <source>  <age>18</age>
<salary>200000000</salary>  <acno>2356376438743</acno>  </source>
```

After the adapter operation acknowledges sending the message to the IR, the job then generates the output file JMSSourceOutput_PutTopic.xml under the directory **<DS_COMMON_DIR>**/adapters/JMS/samples/xml. The contents of the file should be similar to the message received by the sample test with the addition of a timestamp. Note that this file is created as a result of the design of the job, not as a result of the adapter operation sending a reply message to the job.

Testing on UNIX

Run the sample application by running sampleTest_Put.sh file from the command prompt. This sample application listens at the destination queue configured for the Put operation instance.

Execute the batch Job JMSPutOperation_BatchJob from the Designer.

7.7.3.5 Testing Get: Request/Reply

To configure the operation type Get (Request/Reply), enter the following information in the operation instance configuration page in the Web Administrator.

Option	Value
Operation instance	JMSGetOperation

Option	Value
Polling interval	1000
Thread count	1
Enable	true
Source queue	Queue.ActaQueueGet
Service	Queue.TestService
Timeout	2000
Continue after error	true
Default response queue	Queue.ActaReplyQueueGet
Undelivered queue (optional)	Queue.ActaUndeliveredQueue

After entering this information, click *Apply* and restart the JMS Adapter instance. When the JMS Adapter starts running, the operation instance also starts running.

Testing on Windows

Run the sample application (external IR) by running `sampleTest_Send.bat` file from the command prompt. This sample application sends the message at the source queue of the Get operation instance configured in the software.

Also, run another sample application (external IR) by running the batch file `sampleTest_Get.bat` file, which receives the reply from SAP Data Services on a default response queue.

The sample application `sampleTest_Send.bat` (external IR) sends the message as a request on a source queue configured for JMSGetOperation instance. JMSGetOperation instance invokes the real-time batch job and also sends the reply back at the default response queue. The sample application `sampleTest_Get.bat` (external IR) receives the reply on this default response queue. If any error occurs while invoking another service from this Job service, then the original message is sent to the undelivered queue, for reference by the IR.

Testing on UNIX

Run the sample application (external IR) by running `sampleTest_Send.sh` file from the command prompt. This sample application sends the message at the request queue of the operation instance configured in the software.

Also, run another sample application (external IR) by running the batch file `sampleTest_Get.sh` file. This receives the reply from the software on a default response queue.

The sample application `sampleTest_Send.sh` (external IR) sends the message as a request on a source queue configured for JMSGetOperation instance. JMSGetOperation instance will invoke the real-time batch job and also sends the reply back at the default response queue. The sample application `sampleTest_Get.sh` (external IR) receives the reply on this default response queue. If any error occurs while invoking another service from this Job service, then the error message is sent to the undelivered queue, for reference by the IR.

7.7.3.6 Testing GetTopic: Request/Acknowledge

To configure the operation type Get topic (Request/Acknowledge), enter the following information in the operation instance configuration page in the Web Administrator.

Option	Value
Operation instance	JMSGetTopicOperation
Polling interval	1000
Thread count	1
Enable	true
Source topic	Topic.MyTopic
Service	Topic.TestService
Timeout	2000
Continue after error	true

After entering this information, click *Apply* and restart the JMS Adapter instance. When the JMS Adapter starts running, the operation instance also starts running.

Testing on Windows

Run the sample application (external IR) by running the `sampleTest_GetTopic.bat` file from the command prompt. This sample application publishes a message to the source topic of the GetTopic operation instance.

JMSGetTopicOperation, which has subscribed to the topic, receives the message and sends it to the real-time service. The service then puts the message into file `JMSFileTarget_GetTopic.xml` in directory **`<DS_COMMON_DIR>`**`/adapters/jms/samples/xml`.

Testing on UNIX

Run the sample application (external IR) by running the `sampleTest_GetTopic.sh` file from the command prompt. This sample application publishes a message to the source topic of the GetTopic operation instance.

JMSGetTopicOperation, which has subscribed to the topic, receives the message and sends it to the real-time service. The service then puts the message into file `JMSFileTarget_GetTopic.xml` in directory **`<DS_COMMON_DIR>`**`/adapters/jms/samples/xml`.

7.7.3.7 Testing Get: Request/Acknowledge

To configure the operation type Get (Request/Acknowledgment), enter the following information in the operation instance configuration page in the Web Administrator.

Option	Value
Operation instance	JMSGetOperation
Polling interval	1000
Thread count	1
Enable	true
Source queue	Queue.ActaQueueGet
Service	Queue.TestService
Timeout	2000
Continue after error	true
Default response queue	ℹ Note When you specify a value, this operation changes from Request/Acknowledgement to Request/Reply.
Undelivered queue	ℹ Note When you specify a value, this operation changes from Request/Acknowledgement to Request/Reply.

After entering this information, click *Apply* and restart the JMS Adapter instance. When the JMS Adapter starts running, the operation instance also starts running.

Testing on Windows

Run the sample application by running `sampleTest_Send.bat` file from the command prompt.

This sample application (external IR) sends the message as a request on a source queue configured for JMSGetOperation instance. JMSGetOperation instance invokes the real-time batch job. This creates an output file `JMSSourceOutput_Get.xml` as an acknowledgement at the location **`<DS_COMMON_DIR>`**`/adapters/JMS/samples/xml`. No response is sent to the default response queue since it is not configured for this type of operation.

Testing on UNIX

Run the sample application by running `sampleTest_Send.sh` file from the command prompt.

This sample application (external IR) sends the message as a request on a source queue configured for JMSGetOperation instance. JMSGetOperation instance invokes the real-time batch job. This creates an output file `JMSSourceOutput_Get.xml` as an acknowledgement at the location **`<DS_COMMON_DIR>`**`/adapters/JMS/samples/xml`. No response is sent to the default response queue since it is not configured for this type of operation.

7.7.3.8 Testing Put: Request/Acknowledge

To configure the operation type Put (Request/Acknowledge), enter the following information in the operation instance configuration page in the Web Administrator.

Option	Value
Operation instance	JMSPutOperation
Thread count	1
Display name	JMSPutOperation
Description	This operation instance represents the Put Request/ Acknowledge operation. It queues the message to the configured destination queue.
Enable	true
Destination queue	Queue.MyQueue
Request format	**<DS_COMMON_DIR>**/adapters/JMS/samples/dtd/JMSPUT_SOURCE.dtd
Request XML root element	source

Click *Apply* after entering this information, then restart the adapter instance.

When the JMS Adapter is running, the operation instance is also running.

Testing on Windows

Open a command prompt window and change directory to **<DS_COMMON_DIR>**\adapters\jms\samples. Run the sample application (external IR) by running sampleTest_Put.bat. The application should display the message:

```
Ready to receive message from queue Queue.MyQueue.
```

Execute the batch Job JMSPutOperation_BatchJob from the Designer.

The sample application (external IR) listens for a message to arrive at the request queue of the JMSPutOperation instance. When it receives the message, it will print a message to the command prompt window such as:

```
Received message:  <?xml version="1.0" encoding="UTF-8"?>  <source>  <age>18</age>
<salary>200000000</salary>  <acno>2356376438743</acno>  </source>
```

After the adapter operation acknowledges sending the message to the IR, the job then generates the output file JMSSourceOutput_Put.xml under the directory **<DS_COMMON_DIR>**/adapters/JMS/samples/xml. The contents of the file should be similar to the message received by the sample test with the addition of a timestamp. Note that this file is created as a result of the design of the job, not as a result of the adapter operation sending a reply message to the job.

Testing on UNIX

Run the sample application by running `sampleTest_Put.sh` file from the command prompt. This sample application listens at the destination queue configured for the Put operation instance.

Execute the batch Job JMSPutOperation_BatchJob from the Designer.

The sample application receives the message from the destination queue and an output file `JMSSourceOutput_Put.xml` as an acknowledgment gets created under the directory **`<DS_COMMON_DIR>`**`/adapters/JMS/samples/xml`.

7.7.3.9 Technical implementation

7.7.3.9.1 Design considerations

In the current design:

- JMS queues and topics used in the Operation instances must be pre-configured in the Messaging System.
- Only XML messages are handled.
- GetTopic operations should be configured to specify a Thread Count of 1. Since each thread would be a subscriber to the topic, each thread would receive the same message and send it to the service, resulting in multiple copies of the same message going to the service.

7.7.3.9.2 Error handling and tracing

Error messages are logged in error log file under the **`<DS_COMMON_DIR>`**`/adapters/log` directory before throwing any exception. The name of the error log file is same as the name of the adapter configured in the Administrator.

For tracing, the trace messages are logged in the trace file under the **`<DS_COMMON_DIR>`**`/adapters/log` directory. The name of the trace file is same as the name of the adapter configured in the Administrator. You can enable the trace option in the Administrator for this adapter. Trace message shows the execution flow of the adapter and contain useful information on finding the cause of an error. The output in this trace file is of great help for SAP Business User Support.

7.7.4 Appendix

7.7.4.1 Weblogic as JMS provider

Before you run the SAP Data Services Adapter for JMS, you need to create a JMSServer, Connection Factory and configure JMS queues.

- Create a JMS Server
- Start the BEA Weblogic server.
- Open the Weblogic console.
- Under services\JMS, click *Servers*.
- Click Create a new JMS Server button.

Create the instance of JMS server. Then, click *Create*.

Click the Target link on the screen and select the server from available block to a chosen block. Click *Apply* to create the server instance.

7.7.4.1.1 To create a JMS Connection Factory

1. Start the BEA Weblogic server
2. Open the Weblogic console
3. Under services\JMS, click *Connection Factories*.

Configure the Connection Factory. For testing purposes, “JMSConnections.AdapterConnectionFactory” must be configured.

Click the Target link on the screen. Select the server from available block to chosen block.

7.7.4.1.2 To configure the JMS Connection Factory

For testing purposes, “JMSConnections.AdapterConnectionFactory” must be configured.

1. Click the Target link on the screen.
2. Select the server from available block to chosen block.
3. Click *Apply* to create the connection factory.

7.7.4.1.3 To create a JMS queue

1. Start the BEA Weblogic server.
2. Open the Weblogic console
3. Under services\JMS\Servers\ConfigJMSServer\Destinations, click *Create a New JMS Queue*.

For testing purposes, configure the following queues in the server:

- Queue.MyQueue
- Queue.ActaQueuePutGet
- Queue.ActaQueuePutGet1
- Queue.ActaQueueGet
- Queue.ActaReplyQueueGet

- Queue.ActaUndeliveredQueue

7.8 Using the HTTP adapter

7.8.1 Introduction

7.8.1.1 Audience and assumptions

This section assumes the following:

- You understand how to use the Designer to design and run Data Services data flows.
- You have a basic understanding of how to use the Administrator to administer Data Services processes. (To use an adapter, you administer it from the Administrator.)
- You have a working knowledge of the environment this adapter is targeting.
- You know the role an adapter plays in business systems integration
- You have some familiarity with the XML markup language and XML configuration schemas.
- Because you will integrate Data Services and an external system, some familiarity with systems administration and systems integration issues is recommended.

7.8.1.2 About this section

This section tells you how to use the HTTP Adapter for integrating SAP Data Services with external applications using the HTTP protocol.

This section provides a detailed description of installing the HTTP Adapter. This includes the descriptions of the pre-requisite software along with their supported versions, the details of the adapter components, the environment setup for both Data Services and Tomcat and instructions for executing the adapter.

7.8.2 Overview

What does the HTTP Adapter provide?

The HTTP Adapter provides data transfer using HTTP and HTTPS protocols.

What is HTTP protocol?

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990 and its use has increased steadily over the years, mainly because it has proven useful as a generic middleware protocol.

The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a connection with a server.

The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity meta information, and possible entity-body content.

HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80 [19], but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used.

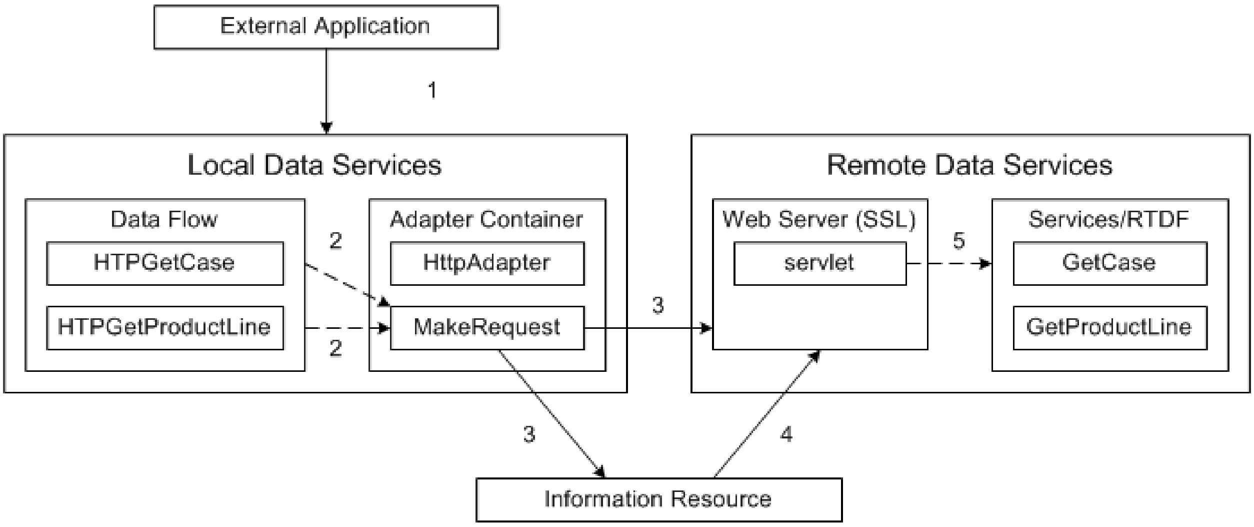
Scope of the HTTP Adapter

The major scope of the HTTP Adapter is:

- To ensure rapid integration of diverse systems and applications by making use of HTTP protocol with the SAP Data Services platform to meet unique business process requirements saving valuable time and effort.
- The HTTP Adapter supports SSL (Secure Socket Layer) to implement security over the HTTP protocol. Using HTTPS protocol, the data will be protected from any unscrupulous elements.
- The HTTP Adapter supports compress type data encoding while sending and receiving the information and thus saves network traffic.
- Request/Reply and Request/Acknowledge services by SAP Data Services can be initiated through this adapter.

7.8.3 Architecture

The following diagram shows the functional overview of the HTTP Adapter. It depicts two remote SAP Data Services installations using the HTTP Adapter to exchange information. The same diagram applies for interaction between SAP Data Services and any other third-party software supporting the HTTP protocol.



The flow of control is explained below. Refer to the above diagram.

1. External application invokes a service on SAP Data Services.
2. A data flow makes a call to the adapter operation instance.
3. The operation instance receives the XML data from RTDF and makes a request on the remote Data Services server. The operation instance forms the request URL by reading its configuration file. The URL contains servlet name and the service name, which are configured as part of the operation instance configuration. In the case of requesting to the information resource, a resource-specific URL will be configured as part of the operation instance configuration.
4. The information resource (for example, Siebel) can make a request on the remote Data Services server by using HTTP or HTTPS protocol. The information resource forms the URL, which contains the servlet name and service name.
5. The servlet runs on the HTTP server (Tomcat) that is a part of Data Services. This HTTP server can be SSL enabled, depending on user requirements. The servlet processes the request to get the service name and XML data. It will then invoke that service running locally in Data Services and send the reply back to the client.

7.8.4 Installation and configuration

The HTTP Adapter is installed with every SAP Data Services Job Server.

You need to create and configure an adapter instance and necessary adapter operations of the HTTP Adapter. Adapter operations identify the integration options available for the configured adapter instance. The HTTP Adapter servlet is also installed while installing the HTTP Adapter.

The operations provided with the HTTP Adapter include the following:

- Request/Reply operation
  This operation is used to execute a remote Data Services service in the Request/Reply mode. It makes the request to the remote Data Services machine where the HTTP Adapter servlet is running and waits for the reply. The service name to be executed can be configured as part of the URL while configuring the operation instance.
- Request/Acknowledge operation

This operation is used to execute a remote Data Services service in the Request/Acknowledge mode. It makes the request to the remote Data Services machine where the HTTP Adapter servlet is running and does not wait for the reply. Instead, it sends acknowledgement if the operation is successful. The service name to be executed can be configured as part of the URL while configuring the operation instance.

7.8.4.1 To configure the HTTP Adapter

To use the HTTP Adapter, create an instance of the adapter together with a real-time data flow (RTDF) or data flow that you define using the Designer. Before using an HTTP Adapter instance, you must first configure it as explained in this section. To configure the HTTP Adapter means to configure one or more instances of the adapter and one or more operation instances for each adapter instance.

All SAP Data Services adapters communicate with Data Services through a designated Adapter Manager Job Server. An adapter must first be installed on the same computer as this Job Server before you can integrate the adapter with Data Services using the Administrator and Designer. After the adapter is installed:

1. Use the Server Manager utility to configure adapter connections with the Adapter Manager Job Server. For details, see the “To configure Job Servers” section of the *Administrator Guide*.
2. Open the Administrator and:
   a) Add at least one instance of the adapter to the Data Services system.
   b) If the adapter instance includes operations, add at least one operation for each adapter instance.
   c) Start the adapter instance.

      Operations are started automatically.
3. Open the Designer and create an adapter datastore. Use metadata accessed through the adapter to create batch and/or real-time jobs. For details, see the “Adapter datastores” section of the *Designer Guide*.

7.8.4.2 To configure an HTTP Adapter instance

Use the Administrator to add an HTTP Adapter to the SAP Data Services system and to edit existing adapter configurations. Until you add the adapter in the Administrator, you cannot run jobs using information from that adapter.

To add an adapter instance in the administrator:

1. Select *Adapter Instances* › *Job Server*.
2. Click the *Configuration* tab.
3. Click *Add*.
4. Select the HTTP Adapter from the list of those available on the Job Server.
5. Enter the required information to create an HTTP Adapter instance.
   For details, see “Adapter instance configuration information”.
6. Click *Apply*.

The Administrator adds the adapter instance to the list of those available to the Data Services system.

7.8.4.2.1 Adapter instance configuration information

Complete the following fields in the Administrator to set up an HTTP Adapter instance in the SAP Data Services system:

Field	Configuration information
Adapter instance name	Enter a unique name that identifies this instance of the HTTP Adapter.
Access Server host	Enter the host ID of the computer running the Access Server that connects to this adapter instance.
Access Server connection port	The Access Server host's message broker port. After you log into the Administrator for this Access Server, select *Configuration* > *Interfaces* to view message broker port information.
Classpath	All adapter Java programs require specific JAR files in the CLASSPATH to use when starting `javaw.exe`. For example: • **`<LINK_DIR>`**`/lib/acta_adapter_sdk.jar` • **`<LINK_DIR>`**`/lib/acta_broker_client.jar` • **`<LINK_DIR>`**`/lib/acta_tool.jar` • **`<LINK_DIR>`**`/ext/lib/xerces.jar` • **`<LINK_DIR>`**`/lib/acta_http_adapter.jar` • **`<LINK_DIR>`**`/lib/jcert.jar` • **`<LINK_DIR>`**`/lib/jnet.jar` • **`<LINK_DIR>`**`/lib/jsse.jar`
Autostart	When set to True, the adapter interface automatically starts when the Administrator starts.
Trace mode	Set this flag to control the number of trace messages the adapter writes. There are two settings: • True: The adapter interface writes additional information messages to help debug problems. • False: The adapter interface writes minimal information messages. The adapter writes trace messages to the **`<adapter_instance_name>`**`_trace.txt` file in the **`<DS_COMMON_DIR>`**`/adapters/logs` directory.
Application command line parameters	Additional command line parameters used for the `javaw.exe` command line and for the adapter itself. See specific adapter documentation for details.
Adapter type name	The name of the adapter used to create this instance. (Read-only)
Adapter version	The version of the adapter used to create this instance. (Read-only)
Adapter class	A name that identifies the adapter class. The name depends on the type of adapter. (Read-only)
Keystore password	Required if requests are made using the HTTPS protocol. If a password is given, it is used to check the integrity of the

Field	Configuration information
	keystore data. Otherwise, the integrity of the keystore is not checked.

7.8.4.3 To configure an operation instance

Use the Administrator to add an operation instance to an adapter instance.

1. Select *Adapter Instance* › *Job Server*
2. Click the *Configuration* tab.
3. Click *Operations* under Dependent Objects.
4. Click *Add* to configure a new operation or click the link of an existing operation to edit its configuration.
5. Select an operation type from the list and click *Apply*. The options that appear on this page depend on the operation-specific design.
6. Complete the operation instance configuration form. For details, see “Operation instance configuration information”.
7. Click *Apply*.

7.8.4.3.1 Operation instance configuration information

Request/Reply operation configuration

Complete the following fields in the Administrator to set up a Request/Reply operation instance:

Field	Configuration information
Operation instance	Enter the unique operation instance name. In the Designer, Request/Reply operation metadata object will be imported with this name.
Thread count	The number of copies of the Request/Reply operation to run in parallel. For parallel (asynchronous) processing of messages coming from a real-time service, use more than one copy. If the sequency of messages is important (synchronous processing), you should not use more than one thread. The default is 1. Note Multiple copies of real-time services must be supported by multiple copies of Request/Reply.
Display name	Enter the operation instance display name. This display name will be visible in the Designer's metadata browsing window.

Field	Configuration information
Description	Enter the operation instance description. This description will be visible in the Designer's metadata browsing window.
Enable	True if the Adapter SDK will start this operation instance when the adapter starts, otherwise false.
Target URL	URL where you want to send the HTTP request. Data Services uses the following server URL format: `http://<host>:<port>/admin/servlet/com.acta.adapter.http.server.HTTPActaServlet?ServiceName=<ServiceName>` - Host: The IP address or host name of the Access Server. - Port: The port number of the Access Server. - ServiceName: The name of the service.
Request method	The HTTP request method to be used for submitting the request. The possible values are POST and GET.
Content-Type	This is used to set the content type header of the request. It specifies the nature of the data by giving type and subtype identifiers.
Content-Language	The ISO code for the language in which the request's document is written. For example, `en` means that the language is English in one of its forms.
Content-Encoding	Specifies the encoding mechanism used for sending the request. Currently only x-compress and x-gzip are used.
Continue if untrusted	Specifies whether to continue the operation if the HTTP server is untrusted when using the HTTPS protocol. - True: The operation instance will continue for untrusted servers. - False: The operation instance will be terminated for untrusted servers.
Request DTD	The DTD file name that defines the request XML message used in this operation.
Request XML root element	The name of the XML root element in the request DTD.
Reply DTD	The DTD file name that defines the reply XML message used in this operation.
Reply XML root element	The name of the XML root element in the reply DTD.

Note

Restart the HTTP Adapter instance so that all configuration changes take effect.

Request/Acknowledge operation configuration

Complete the following fields in the Administrator to configure a Request/Acknowledge operation instance:

Field	Configuration information
Operation instance	Enter the unique operation instance name. In the Designer, Request/Acknowledge operation metadata object will be imported with this name.
Thread count	The number of copies of the Request/Acknowledge operation to run in parallel. For parallel (asynchronous) processing of messages coming from a real-time service, use more than one copy. If the sequency of messages is important (synchronous processing), you should not use more than one thread. The default is 1. Note Multiple copies of real-time services must be supported by multiple copies of Request/Acknowledge.
Display name	Enter the operation instance display name. This display name will be visible in the Designer's metadata browsing window.
Description	Enter the operation instance description. This description will be visible in the Designer's metadata browsing window.
Enable	True if the Adapter SDK will start this operation instance when the adapter starts, otherwise false.
Target URL	URL where you want to send the HTTP request. Data Services uses the following server URL format: `http://<host>:<port>/admin/servlet/com.acta.adapter.http.server.HTTPActaServlet?ServiceName=<ServiceName>` ● Host: The IP address or host name of the Access Server. ● Port: The port number of the Access Server. ● ServiceName: The name of the service.
Request method	The HTTP request method to be used for submitting the request. The possible values are POST and GET.
Content-Type	This is used to set the content type header of the request. It specifies the nature of the data by giving type and subtype identifiers.
Content-Language	The ISO code for the language in which the request document is written. For example, `en` means that the language is English in one of its forms.
Content-Encoding	Specifies the encoding mechanism used for sending the request. Currently only x-compress and x-gzip are used.
Continue if untrusted	Specifies whether to continue the operation if the HTTP server is untrusted when using the HTTPS protocol. ● True: The operation instance will continue for untrusted servers. ● False: The operation instance will be terminated for untrusted servers.

Field	Configuration information
Request DTD	The DTD file name that defines the request XML message used in this operation.
Request XML root element	The name of the XML root element in the request DTD.

i Note

Restart the HTTP Adapter instance so that all configuration changes take effect.

7.8.4.4 Defining the adapter datastore

Use the HTTP Adapter with an RTDF/data flow when the the RTDF/data flow passes a message to an operation instance, using either:

- An outbound message (for Request/Acknowledge operations)
- A message function (for Request/Reply operations)

You must define an adapter datastore in the Designer. An RTDF/data flow can then pass a message to one of the adapter's operation instances defined in the datastore.

To define an adapter datastore, you must:

- Define a datastore object for each adapter instance.
- Define one function or one outbound message for each operation instance to which you want to pass a message.

The following sections summarize the Designer tasks involved. For complete information, see the *Designer Guide*.

7.8.4.4.1 To define an adapter datastore

For each adapter instance, define a corresponding datastore object in the Designer object library, in the *Datastore Editor* window.

1. In the *Job Server* box, select the Job server that is configured to handle your HTTP adapter.
2. For *Adapter instance name*, choose the instance name you configured in the Administrator.
3. Click to select the Adapter Properties tab and enter values for each property.
4. Click *OK* to save values and create the datastore.

7.8.4.4.2 To import message functions and outbound messages

Messages can be passed from a real-time data flow (RTDF) to an operation instance. You must import either a function or an outbound message (depending on the type of operation involved) in the Designer Datastore library for each operation instance.

Real-time data flows use:

Method	Description
Message functions	Pass messages to an operation instance if the RTDF waits for a return XML message from the information resource.
Outbound messages	Outbound messages pass messages to an operation instance if the RTDF waits for a confirmation only (not a return XML message) from the information resource.

The operation types in the HTTP Adapter have the following invocation types:

Operation type	Invocation type
Request/Reply	Message function
Request/Acknowledge	Outbound message

To import the message functions and outbound messages:

1. In the Designer, double click on the datastore that has an associated HTTP Adapter instance.
2. In the *Adapter Metadata Browser* window, right-click on the operation instance to import and select *Import*.

The selected operation instance will be added to the datastore. These message functions and outbound messages can be used for creating the RTDF/data flow in Data Services.

7.8.4.5 Configuring SSL with the HTTP adapter

With Secure Sockets Layer (SSL), the HTTP Adapter can use secure transport over the TCP/IP network.

Server-side configuration

To use SSL with the HTTP adapter, you must properly configure your web application server for SSL support.

If you are using the default web application server bundled with SAP BusinessObjects BI platform, see the *SAP BusinessObjects BI Platform Administrator Guide* for information on how to configure SSL.

If you are using a different third-party web application server, see the web application server's documentation.

Client-side configuration

On the client side, the HTTP Adapter client internally handles the details of certificate authentication by implementin the X509TrustManager interface and using SSLSocketFactory classes from the HttpsURLConnection class.

Whenever an HTTPS request is made to the SSL-enabled web server, the client requests the server's certificate, which may be issued by a standard authority, such as VeriSign. If the HTTP client finds the certificate to be one that is trusted by comparing it to the certificate store in **<LINK_DIR>**`/ext/jre/lib/security`, it retrieves all data from the web server. In the case of an un-trusted certificate, the HTTP client throws an SSLException to the caller.

The HTTP client requires the password for querying the local keystore for verification. This password can be supplied through the `keystorePassword` parameter specified as a part of the adapter configuration.

The operation instance will read the configurable *Continue if untrusted* flag and, based on its value, trust the unknown server and its certificate. If the parameter is set to **`False`**, then the SSLException is shown to the user with a friendly message and logged in Data Services' trace files and the client does not retrieve any data from the server. If the parameter is set to **`True`**, then the SSLException is logged in Data Services' error and trace files and the client proceeds to retrieve data from the server. The certificate file `untrust.cer` is downloaded to the user's current working directory or to the **`<LINK_DIR>`**`/bin` directory. This certificate file can later be imported into the JDK certificate keystore by using the `keytool` command-line utility:

```
keytool -import -alias <description> -file untrust.cer -keystore
<full_path_of_cacerts_file> cacerts -storepass changeit
```

7.8.5 Using the HTTP Adapter

To start the adapter instance:

1. Select the *Status* tab under *Adapter Instances* in the Administrator.
2. Check the *Select* box next to the adapter instance that you configured previously.
3. Click *Start*.

When your adapter instance and its operations start, the message “Started” appears in the status column. To confirm that all operations are started, click *Operations* in the *Dependent Objects* column.

If you have a real-time service set up on your system, you can invoke it through the HTTP interface:

`http://localhost:8080/admin/jsp/InvokeService.jsp`

Using this interface, you can invoke the selected service by sending the input XML to the HTTP Adapter servlet running on the remote machine where the service is configured.

> Note
>
> For information about how to set up a test service, see the “Verifying real-time connectivity” section in the *Installation Guide*.

Request/Reply from Data Services

The Request/Reply operation instance sends the request to the remote SAP Data Services machine and waits for the reply.

To check the sample Request/Reply operation, see the “Testing the Request/Reply operation” section.

Request/Acknowledge from Data Services

The Request/Acknowledge operation instance sends the message to the remote SAP Data Services machine and gives an acknowledgement.

To check the sample Request/Acknowledge operation, see the "Testing the Request/Acknowledge operation" section.

7.8.5.1 Testing the Request/Reply operation

To configure the Request/Reply operation for testing, use the following information to configure the operation instance in the Administrator:

Field	Configuration information
Operation instance	`HTTP_ReqReply_Function`
Thread count	`1`
Display name	`HTTP_ReqReply_Function`
Description	`Performs the Request/Reply operation`
Enable	`true`

Target URL	For HTTP operation, use:
	`http://<ds_host_name>:<access_server_port>/admin/servlet/com.acta.adapter.http.server.HTTPActaServlet?ServiceName=Test`
	For HTTPS operation, use:
	`https://<ds_host_name>:<tomcat_https_port>/admin/servlet/com.acta.adapter.http.server.HTTPActaServlet?ServiceName=Test`
	Note
	By default, the HTTPS port of the Tomcat server is 8443. This can be changed in the Tomcat configuration file (`acta-server.xml` on Windows, and `acta-server1.xml` on UNIX)

Request method	`Post`
Content-Type	`text/xml`
Content-Language	`en`

Field	Configuration information
Content-Encoding	**application/nocompress**
Continue if untrusted	**true**
Request DTD	**<DS_COMMON_DIR>**/adapters/Http/samples/dtd/HTTPTestIn.dtd
Request XML root element	**test**
Reply DTD	**<DS_COMMON_DIR>**/adapters/Http/samples/dtd/HTTPTestOut.dtd
Reply XML root element	**test**

After configuring the operation instance, click *Apply* and then restart the HTTP Adapter instance. After the HTTP Adapter is running, the operation instance will also be in a running state.

Execute the HTTP_ReqRep_BatchJob job in the Designer.

After the batch job executes successfully, an output file OutputRep.xml will be created in the **<DS_COMMON_DIR>**/adapters/Http/samples/xml directory.

7.8.5.2 Testing the Request/Acknowledge operation

To configure the Request/Acknowledge operation for testing, use the following information to configure the operation instance in the Administrator:

Field	Configuration information
Operation instance	**HTTP_ReqAck_Outbound**
Thread count	**1**
Display name	**HTTP_ReqAck_Outbound**
Description	**Performs the Request/Acknowledge operation**
Enable	**true**
Target URL	For HTTP operation, use: **http://<ds_host_name>:<access_server_port>/admin/servlet/com.acta.adapter.http.server.HTTPActaServlet?ServiceName=Test** For HTTPS operation, use: **https://<ds_host_name>:<tomcat_https_port>/admin/servlet/**

Field	Configuration information
	com.acta.adapter.http.server.HTTPActaServlet?ServiceName=Test Note By default, the HTTPS port of the Tomcat server is 8443. This can be changed in the Tomcat configuration file (acta-server.xml on Windows, and acta-server1.xml on UNIX)
Request method	**Post**
Content-Type	**text/xml**
Content-Language	**en**
Content-Encoding	**application/nocompress**
Continue if untrusted	**true**
Request DTD	**<DS_COMMON_DIR>**/adapters/Http/samples/dtd/HTTPTestIn.dtd
Request XML root element	**test**

After configuring the operation instance, click *Apply* and then restart the HTTP Adapter instance. After the HTTP Adapter is running, the operation instance will also be in a running state.

Execute the HTTP_ReqAck_BatchJob job in the Designer.

After the batch job executes successfully, an output file OutputAck.xml will be created in the **<DS_COMMON_DIR>**/adapters/Http/samples/xml directory.

7.8.6 Error handling and tracing

All error and trace messages are logged to the log files in the **<DS_COMMON_DIR>**/adapters/log directory. The names of the error and trace log files match the names of the adapter instance as configured in the Administrator, and appended with _error.txt for error logs and _trace.txt for trace logs.

For example, if the name of the HTTP Adapter instance is “HTTPAdapter”, the name of the error file will be HTTPAdapter_error.txt and the name of the trace file will be HTTPAdapter_trace.txt.

7.9 Object creation XML toolkit

7.9.1 Overview

The object creation XML toolkit is a collection of utilities and features that enable you to programmatically create objects such as jobs, data flows, and work flows from your own application and then import, validate, and execute them in SAP Data Services.

The toolkit consists of several primary components:

Component	Purpose
XML schema	Used to defines Data Services objects
Designer import and export tools	Used to create template or example XML of objects
Web services	Used to send externally generated objects to Data Services and import, validate, or execute them
Auxiliary utilities	Used to perform specific tasks required by some jobs, such as datastore password encryption

7.9.2 Using the toolkit

In general, the object creation XML toolkit is suitable for two scenarios:

- Exporting Designer-created object definitions for use as templates that are customized with substitution variables
- Exporting Designer-created object definitions for use as a guideline for creating new object definitions from scratch

For both scenarios, we recommend that you use the Designer and its import and export capabilities heavily until you have the object definition to generate from within your own application. After you have suitable object defintions, you can use web services to import, validate, and execute jobs within Data Services.

The recommended work flow for using the toolkit is:

1. Create templates of your objects in the Designer.
2. Export your objects from the Designer to the XML format.
3. Customize the exported XML objects for generation in your external application.
4. Import, validate, and execute the XML objects generated by your application through Data Services' web services.

7.9.2.1 Templating objects

A key feature of the object creation XML toolkit is the ability to use the Designer to template your objects. You can use the Designer to create and debug objects, and then export them to an XML format for customization and use in your own external application. To create a template object for use in the object creation XML toolkit, the process is the same as creating any other normal object in the Designer.

> ➜ Tip
>
> Although it is possible to use only the XML schema to write complete repository objects from scratch, we strongly recommend that you use the Designer to create your objects due to complex interactions between many parts of the XML structure.

For more information about using the Designer to create and configure jobs, data flows, transforms, and other objects, see the *Designer Guide*.

For more information about configuration options available for specific objects, see the *Reference Guide*.

7.9.2.2 Exporting objects

The Designer allows you to export SAP Data Services objects in a standardized, reproducible XML format. When exporting objects, you can choose to export an entire job or individual objects. Each export option produces a single XML file that contains all exported objects. If you want to create separate XML files for each object, use the export editor to individually export each object.

For more information about using the export editor, see the “Export/Import” section of the *Administrator Guide*.

7.9.2.3 Adapting objects

Because the options of many XML objects, such as transforms, have complex interactions and do not always match the way the configuration appears in the Designer, it's strongly recommended that you adapt exported versions of these objects that have already been mostly configured how you want. By adapting existing objects, you can be more confident that the objects generated by your application will be free of problems, and function as you expect.

For example, you can configure several transforms using the Designer and then export them to XML files. In your application, you can assemble the pre-configured transforms into working data flows, and use web services to import and run them with Data Services.

Exported objects can still be customized through the use of parameters, as well as simply modifying the XML directly.

Related Information

Parameters and variables [page 2339]

7.9.2.4 Using web services

After generating custom objects in your application, you can use web service operations provided with SAP Data Services to import, validate, and execute them.

The general process for using the object creation XML toolkit with web service operations has several steps:

1. Log in

   If web service security is enabled, use the Logon operation and get a session object.
2. Import objects

   Use the Import_Repo_Object operation to import objects to the repository. You can import single or multiple objects at a time. However, for large XML, you may need to import objects one at a time.

   If an error occurs during importing, the operation returns the error message.
3. Validate objects

   After the objects have been imported successfully, use the Validate_Repo_Object to perform a semantic validation. You can also perform the validation on only the highest level object to recursively validate all dependent objects.

   If an error occurs during validation, the operation returns the error message.
4. Execute objects

   After the objects have been validated successfully, use the Run_Batch_Job operation to execute the job now stored in the repository. Use the job name as the parameter for the execution request.

   If an error occurs during execution, use the returned runID code and the Get_Error_Log and Get_Trace_Log operations to retrieve messages specific to this execution of the job.
5. Retrieve operational metadata

   Use the runID code returned by Run_Batch_Job along with the Get_Error_Log, Get_Trace_Log, and Get_Monitor_Log operations to retrieve errors, warnings, trace messages, and performance statistics for this execution of the job.
6. Remove objects (optional)

   If you don't want the objects to persist in the repository after execution, use the Delete_Repo_Objects operation to remove them. Because the operation does not remove dependencies automatically, you need to call it once for each object you want to remove.
7. Remove operational metadata

   Error and trace logs for non-execution requests are automatically cleared at the completion of each request. However, execution logs are not automatically cleared, and must be cleaned up manually. You can schedule the cleanup of execution logs by setting an appropriate log retention period in the Administrator.

   For more information, see the *Management Console Guide*.
8. Log out

   If you used the Login operation to take advantage of web services security, use the Logout operation to log out and end the session.

Related Information

Logon [page 2226]

Import_Repo_Object [page 2250]
Validate_Repo_Object [page 2251]
Run_Batch_Job [page 2244]
Get_Error_Log [page 2241]
Get_Trace_Log [page 2243]
Get_Monitor_Log [page 2243]
Delete_Repo_Objects [page 2248]
Logout [page 2227]

7.9.2.5 Encrypting passwords

For security, passwords for things such as datastore connections are encrypted when stored in the repository. However, when importing objects, SAP Data Services does not perform the encryption operation. Because of this, in order to use passwords in externally generated objects, you must use the encrypted form of the password in the generated XML.

To encrypt passwords outside of Data Services, use the `al_encrypt` utility included with the object creation XML toolkit. You can use `al_encrypt` with parameters for either an existing keyfile or a plain-text passphrase.

By default, `al_encrypt` outputs the encrypted password to the screen. However, you can use output redirection to store the encrypted password to a file, and then read the password from that file when generating the XML for your object. For example:

```
al_encrypt -e mypassword -p thepassphrase > c:\password.txt
```

For complete syntax information for `al_encrypt`, see the *Administrator Guide*: Command line administration, Password encryption.

7.9.2.6 Best practices

When using the object creation XML toolkit, there are a number of best practices that you can follow to simplify your work flow and minimize any problems that you may encounter.

Importing objects

- While you are allowed to import multiple objects at once, you may need to import objects individually when the XML syntax is large.
- Objects should be imported into the repository in order. That is, lower level objects in the dependency chain should be imported before higher level objects. For example, if you have a job that contains a data flow that uses a file format, you should import the file format, followed by the data flow, and then the job. By properly maintaining the correct import order, you can avoid cross-referencing issues.
- You can avoid import problems by validating your generated XML before importing. That is, ensuring that elements are closed correctly, and so on.

Validating objects

- Validate your objects using the Designer during your design phase. The validation web service performs only a runtime validation that is not as comprehensive as the Designer's validation. Validating in the Designer can provide you with more detailed information that may be helpful in resolving issues.
- You can validate objects individually or recursively by validating a high level object.

Other

- All object definitions must be placed within the correct container element, DataIntegratorExport. You can place one or more object definitions into the DataIntegratorExport element.

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  <DataIntegratorExport repositoryVersion="12.2.0.0000" productVersion="12.1.0.0">

     <!-- One or more object definitions -->

  </DataIntegratorExport>
  ```
- Some Data Services objects support expressions. For example, you can assign a query to the output field of a query transform. In addition, scripts and custom functions are defined as expressions. The expressions used in these objects are represented twice in exported XML:

  1. As an expr attribute with the same format as shown in the Designer
  2. As a complex XML hierarchy

  When importing these objects, only the format contained in the expr attribute is required.
- For Data Quality transform custom configurations, it is not required to have best practice input and output fields defined.

7.9.2.7 Limitations

General limitations

You should be aware of several general limitations when using the object creation XML toolkit:

- Some parts of object validation can only be performed in the Designer user interface.
  To more quickly identify and fix validation problems, you can import and validate your objects in the Designer when you are developing custom objects.
- The web service operations do not support import or bulk loading of Data Cleanse dictionaries.
  Before using other web services with the Data Cleanse transform, you must configure the appropriate dictionaries with the Designer.

Concurrent use issues

When multiple users access the same repository concurrently, the Data Services engine locks appropriate tables and serializes requests so that the repository is always in a structurally valid state. However, you should be aware of other issues that can arise due to concurrent repository access:

- Referential integrity violation due to object removal
  The Delete_Repo_Objects web service operation does not enforce the rule that an object cannot be deleted if it is being used by other objects. The exception is a datastore, which cannot be deleted if it contains any child objects.
  You should maintain referential object integrity in your application outside Data Services.
- Interference between web service operations
  Some available web service operations have the potential to interfere with others if they are currently running. For example, a job could fail if the job is deleted with the Delete_Repo_Objects operation at the same time it is being executed with the Run_Batch_Job operation.
  You should be careful to synchronize web service operations so that conflicts do not occur.

One way to avoid most concurrent use issues is by maintaining good naming conventions. For example, by using unique names for objects created in different instances of your application, you can effectively create a segregated domain for each application instance. As a result, the application instances will not interfere with each other when performing operations such as creating, modifying, or reading repository objects.

7.9.3 XML schema reference

The object creation XML toolkit supports the creation and usage of all objects available in Data Services. This section provides information about the structure and usage of some of the most common objects. For objects not covered in detail, use the export function of the Designer and the “Objects” section in the *Reference Guide* as a guide for creating and using your own objects.

Caution

All examples provided in this section are for reference only. Do not attempt to run them.

Enclosing objects to import

To import XML content into the Data Services repository, it must be enclosed in a DataIntegratorExport element. Specify the repository and product versions using the repositoryVersion and productVersion attributes.

```
<?xml version="1.0" encoding="UTF-8" ?>
<DataIntegratorExport repositoryVersion="12.2.0.0000"
  productVersion="12.2.0.0">

</DataIntegratorExport>
```

The DataIntegratorExport element is required only once per XML file, and should contain all objects that you want to import.

7.9.3.1 Batch job

➜ Tip

It's strongly recommended that you use the Designer to create and export your objects, and then use your application to adapt them as required. Many object options and parameters interact in complex ways, and do not always match the way they appear in the Designer.

Creating the object definition

Open the batch job definition with the DIJob element.

```
<DIJob name="<JobName>" typeId="2">
```

Specify steps within the job using the DISteps and DICallStep elements.

```
<DISteps>
  <DICallStep typeID="1" calledObjectType="Dataflow"
    name="<DataflowName>"></DICallStep>
</DISteps>
```

Apply job-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="job_checkpoint_enabled" value="no" />
  <DIAttribute name="job_collect_statistics" value="no" />
  <DIAttribute name="job_collect_statistics_monitor"
    value="no" />
  <DIAttribute name="job_enable_assemblers" value="yes" />
  ...
</DIAttributes>
```

End the job definition with the closing tag of the DIJob element.

```
</DIJob>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording used in the Designer. For a complete description of each attribute, see the “Descriptions of objects” section in the *Designer Guide*.

DIAttribute name	Designer
job_checkpoint_enabled	
job_collect_statistics	▶ *Job properties* ▶ *Execution options* ▶ *Collect statistics for optimization* ▶

DIAttribute name	Designer
job_collect_statistics_monitor	*Job properties* > *Execution options* > *Collect statistics for monitoring*
job_enable_assemblers	
job_enable_audit	
job_enable_dataquality	
job_export_repo	
job_export_reports	
job_isrecoverable	
job_mode	
job_monitor_sample_rate	*Job properties* > *Execution options* > *Monitor sample rate*
job_name	*Job properties* > *General* > *Name*
job_print_version	
job_testmode_enabled	
job_trace_abapquery	*Job properties* > *Trace* > *Trace ABAP Query*
job_trace_all	*Job properties* > *Execution options* > *Print all trace messages*
job_trace_ascomm	*Job properties* > *Trace* > *Access Server Communication*
job_trace_assemblers	*Job properties* > *Trace* > *Assemblers*
job_trace_audit	*Job properties* > *Trace* > *Audit Data*
job_trace_dataflow	*Job properties* > *Trace* > *Data Flow*
job_trace_idoc_file	*Job properties* > *Trace* > *IDoc file reader*
job_trace_memory_loader	*Job properties* > *Trace* > *Memory Target*
job_trace_memory_reader	*Job properties* > *Trace* > *Memory Source*
job_trace_optimized_dataflow	*Job properties* > *Trace* > *Optimized data flow*
job_trace_parallel_execution	*Job properties* > *Trace* > *Trace Parallel Execution*
job_trace_rfc_function	*Job properties* > *Trace* > *RFC Function*
job_trace_row	*Job properties* > *Trace* > *Row*
job_trace_script	*Job properties* > *Trace* > *Scripts and Script Functions*
job_trace_session	*Job properties* > *Trace* > *Session*
job_trace_sql_only	*Job properties* > *Trace* > *SQL Only*

DIAttribute name	Designer
job_trace_sqlfunctions	*Job properties* > *Trace* > *SQL Functions*
job_trace_sqlloaders	*Job properties* > *Trace* > *SQL Loaders*
job_trace_sqlreaders	*Job properties* > *Trace* > *SQL Readers*
job_trace_sqltransforms	*Job properties* > *Trace* > *SQL Transforms*
job_trace_stored_procedure	*Job properties* > *Trace* > *Stored Procedure*
job_trace_table	*Job properties* > *Trace* > *Tables*
job_trace_table_reader	*Job properties* > *Trace*
job_trace_transform	*Job properties* > *Trace* > *Transform*
job_trace_userfunction	*Job properties* > *Trace*
job_trace_usertransform	*Job properties* > *Trace*
job_trace_workflow	*Job properties* > *Trace* > *Work Flow*
job_type	
job_use_statistics	*Job properties* > *Execution options* > *Use collected statistics*
locale_codepage	
locale_language	
locale_territory	

7.9.3.2 work flow

➜ Tip

It's strongly recommended that you use the Designer to create and export your objects, and then use your application to adapt them as required. Many object options and parameters interact in complex ways, and do not always match the way they appear in the Designer.

Creating the object definition

Open the work flow definition with the DIWorkflow element.

```
<DIWorkflow name="<WorkflowName>" typeId="2">
```

Specify steps within the work flow using the DISteps, DICallStep, and DIScript elements.

```
<DISteps>
  <DIScript>
```

```
    <DIUIOptions>
      <DIAttribute name="ui_display_name"
        value="<ScriptName>" />
    </DUIOptions>
    ...
  </DIScript>
  <DICallStep typeID="1" calledObjectType="Dataflow"
    name="<DataflowName>"></DICallStep>
  <DIScript>
    <DIUIOptions>
      <DIAttribute name="ui_display_name"
        value="<ScriptName>" />
    </DUIOptions>
    ...
  </DIScript>
</DISteps>
```

Apply workflow-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="run_once" value="no" />
  <DIAttribute name="unit_of_recovery" value="no" />
</DIAttributes>
```

End the work flow definition with the closing tag of the DIWorkflow element.

```
</DIWorkflow>
```

Available DIAttribute names

The available DIAttribute names correspond to work flow parameters and properties, but do not necessarily match the wording used in the Designer. For a complete description of each attribute, see the “Descriptions of objects” section in the *Designer Guide*.

DIAttribute name	Designer
run_once	*Properties* > *General* > *Execute only once*
unit_of_recovery	*Properties* > *General* > *Recover as a unit*

7.9.3.3 data flow

➡ Tip

It's strongly recommended that you use the Designer to create and export your objects, and then use your application to adapt them as required. Many object options and parameters interact in complex ways, and do not always match the way they appear in the Designer.

Creating the object definition

Open the data flow definition with the DIDataflow element.

```
<DIDataflow name="<DataflowName>" typeId="1">
```

Invoke transforms within DIDataflow by using the DITransforms element.

```
<DITransforms>
  <DIFileSource typeId="33" formatName="<FileFormatName>"
    filename="<filename.txt>"/>
  <DIQuery typeId="22"/>
  <DIFileTarget typeId="3"/>
  ...
</DITransforms>
```

Apply dataflow-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="Cache_type" value="pageable_cache" />
  <DIAttribute name="Parallelism_degree" value="0" />
  <DIAttribute name="allows_both_input_and_output"
    value="yes" />
  <DIAttribute name="run_once" value="no" />
  <DIAttribute name="use_dataflow_links" value="no" />
  <DIAttribute name="use_datastore_links" value="yes" />
  <DIAttribute name="validation_xform_exists" value="no" />
  <DIAttribute name="validation_xform_stats" value="no" />
</DIAttributes>
```

End the data flow definition with the closing tag of the DIDataflow element.

```
</DIDataflow>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording used in the Designer. For a complete description of each attribute, see the “Descriptions of objects” section in the *Designer Guide*.

DIAttribute name	Designer
Cache_type	*Properties* > *General* > *Cache type*
Parallelism_degree	*Properties* > *General* > *Degree of parallelism*
allows_both_input_and_output	
run_once	*Properties* > *General* > *Execute only once*
use_dataflow_links	
use_datastore_links	*Properties* > *General* > *Use database links*
validation_xform_exists	

DIAttribute name	Designer
validation_xform_stats	

7.9.3.4 Script

🡆 Tip

It's strongly recommended that you use the Designer to create and export your objects, and then use your application to adapt them as required. Many object options and parameters interact in complex ways, and do not always match the way they appear in the Designer.

Creating the object definition

Open the script definition with the DIScript element.

```
<DIScript>
```

Specify the name of the script with a DIAttribute element in the DIOptions block.

```
<DIOptions>
  <DIAttribute name="ui_display_name" value="<script_name>" />
</DIOptions>
```

Include expressions with the DIExpression element. In exported objects, additional XML syntax may be present. However, this additional syntax is optional; only the expr attribute of DIExpression is required.

```
<DIExpression isString="true" expr="<script_expression>">
  <<Additional optional XML syntax>/>
</DIExpression>
```

Include functions with the DIFunctionCallStep element. Define the function using a DIExpression element. In exported objects, additional XML syntax may be present. However, this additional syntax is optional.

```
<DIFunctionCallStep typeId="23">
  <DIExpression isString="true" expr="<function_call>">
    <<Additional optional XML syntax>/>
        </DIExpression>
</DIFunctionCallStep>
```

End the script definition with the closing tag of the DIScript element.

```
</DIScript>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording or location used in the Designer. For a complete description of each attribute, see the "Descriptions of objects" section in the *Designer Guide*.

DIAttribute name	Designer
ui_display_name	*Properties* › *General* › *Name*

For more information about scripts, see the "Objects, Script" section in the *Reference Guide*.

7.9.3.5 File format

Tip

It's strongly recommended that you use the Designer to create and export your objects, and then use your application to adapt them as required. Many object options and parameters interact in complex ways, and do not always match the way they appear in the Designer.

Creating the object definition

Open the file format definition with the DIFlatFileDatastore element.

```
<DIFlatFileDatastore name="<FileFormatName>" typeId="3">
```

Specify field names, types, and sizes with DIElement elements in the DISchema block.

```
<DISchema>
  <DIElement name="<FieldName1>" datatype="VARCHAR" size="7">
  </DIElement>
  <DIElement name="<FieldName2>" datatype="VARCHAR" size="7">
  </DIElement>
  ...
</DISchema>
```

Apply file format-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="abap_file_format" value="no" />
  <DIAttribute name="blank_pad" value="leading" />
  <DIAttribute name="cache" value="yes" />
  <DIAttribute name="column_delimiter" value=", " />
  ...
</DIAttributes>
```

End the file format definition with the closing tag of the DIFlatFileDatastore element.

```
</DIFlatFileDatastore>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording used in the Designer. For a complete description of each attribute, see the "Descriptions of objects" section in the *Designer Guide*.

DIAttribute name	Designer
abap_file_format	*File Format Editor > General > Type > SAP transport*
adaptable	*File Format Editor > General > Adaptable Schema*
beginning_of_file_string	*File Format Editor > Input/Output > BOF Marker*
blank_pad	*File Format Editor > Default format > Blank padding*
cache	
column_delimiter	*File Format Editor > Delimiters > Column*
column_width	
column_width**<n>**	*File Format Editor > Field Size* for column **<n>**
data_alignment	*File Format Editor > General > Data Alignment*
date_format	*File Format Editor > Default Format > Date*
datetime_format	*File Format Editor > Default Format > Date-Time*
end_of_file_string	*File Format Editor > Input/Output > EOF Marker*
escape_character	*File Format Editor > Default Format > Escape char*
file_format	
file_location	*File Format Editor > Data File(s) > Location*
file_name	*File Format Editor > Data File(s) > File name(s)*
file_type	*File Format Editor > General > Type*
ignore_row_markers	*File Format Editor > Default Format > Ignore row marker(s)*
locale_codepage	*File Format Editor > Locale > Code page*
locale_language	*File Format Editor > Locale > Language*
locale_territory	
name	*File Format Editor > General > Name*
null_indicator	*File Format Editor > Default Format > NULL inicator*
number_of_rows_to_skip	*File Format Editor > Input/Output > Skipped rows*
number_of_threads	*File Format Editor > General > Parallel process threads*

DIAttribute name	Designer
reader_capture_data_conversion_errors	*File Format Editor* ▸ *Error handling* ▸ *Capture data conversion errors*
reader_capture_row_format_errors	*File Format Editor* ▸ *Error handling* ▸ *Capture row format errors*
reader_error_file_name	*File Format Editor* ▸ *Error handling* ▸ *Error file name*
reader_error_file_root_dir	*File Format Editor* ▸ *Error handling* ▸ *Error file root directory*
reader_log_data_conversion_warnings	*File Format Editor* ▸ *Error handling* ▸ *Log data conversion warnings*
reader_log_row_format_warnings	*File Format Editor* ▸ *Error handling* ▸ *Log row format warnings*
reader_maximum_warnings_to_log	*File Format Editor* ▸ *Error handling* ▸ *Maximum warnings to log*
reader_write_error_rows_to_file	*File Format Editor* ▸ *Error handling* ▸ *Write error rows to file*
root_dir	*File Format Editor* ▸ *Data File(s)* ▸ *Root directory*
row_delimiter	*File Format Editor* ▸ *Delimiters* ▸ *Row*
skip_row_header	*File Format Editor* ▸ *Input/Output* ▸ *Skip row header*
table_weight	
time_format	*File Format Editor* ▸ *Default Format* ▸ *Time*
transfer_argument	*File Format Editor* ▸ *Custom Transfer* ▸ *Arguments*
transfer_custom	*File Format Editor* ▸ *General* ▸ *Custom transfer program*
transfer_name	*File Format Editor* ▸ *Custom Transfer* ▸ *Program executable*
transfer_password	*File Format Editor* ▸ *Custom Transfer* ▸ *Password*
transfer_user	*File Format Editor* ▸ *Custom Transfer* ▸ *User name*
use_root_dir	
write_bom	*File Format Editor* ▸ *Input/Output* ▸ *Write BOM*
write_row_header	*File Format Editor* ▸ *Input/Output* ▸ *Write row header*

7.9.3.5.1 To use as a source

To use a file format as a source within a data flow, invoke the format with the DIFileSource element. The DIFileSource element must be placed within the DITransforms section of a data flow.

```
<DIFileSource typeId="33" formatName="<FormatName>"
  filename="<FileName>">
```

Define the name of the source with a DIAttribute element.

```
<DIUIOptions>
<DIAttribute name="ui_display_name" value="<SourceName>" />
</DIUIOptions>
```

Specify a name for the output schema with the DIOutputView element. By default, it is set to the format name. However, you can change it to any unique string, and use it in all downstream transforms.

```
<DIOutputView name="<SchemaName>" />
```

Apply source-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="adaptable" value="no" />
  <DIAttribute name="cache" value="yes" />
  <DIAttribute name="connection_port" value="no" />
  <DIAttribute name="file_location" value="local" />
  ...
</DIAttributes>
```

End the source definition, with the closing tag of the DIFileSource element.

```
</DIFileSource>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording or location used in the Designer. For a complete description of each attribute, see the “Descriptions of objects” section in the *Designer Guide*.

DIAttribute name	Designer
adaptable	▶ *Source File Editor* ▶ *General* ▶ *Adaptable Schema* ◢
cache	▶ *Source File Editor* ▶ *General* ▶ *Cache* ◢
connection_port	
file_location	▶ *Source File Editor* ▶ *Data File(s)* ▶ *Location* ◢
name	▶ *Source File Editor* ▶ *General* ▶ *Name* ◢
reader_capture_data_conversion_errors	▶ *Source File Editor* ▶ *Error handling* ▶ *Capture data conversion errors* ◢

DIAttribute name	Designer
reader_capture_row_format_errors	*Source File Editor › Error handling › Capture row format errors*
reader_filename_col	*Source File Editor › Source information › Column name*
reader_filename_col_size	*Source File Editor › Source information › Column size*
reader_filename_only	*Source File Editor › Source information › Include path*
reader_include_filename	*Source File Editor › Source information › Include file name column*
reader_log_data_conversion_warnings	*Source File Editor › Error handling › Log data conversion warnings*
reader_log_row_format_warnings	*Source File Editor › Error handling › Log row format warnings*
reader_maximum_warnings_to_log	*Source File Editor › Error handling › Maximum warnings to log*
reader_write_error_rows_to_file	*Source File Editor › Error handling › Write error rows to file*
root_dir	*Source File Editor › Data File(s) › Root directory*
table_weight	
transfer_custom	*Source File Editor › General › Custom transfer program*

7.9.3.5.2 To use as a target

To use a file format as a target within a data flow, invoke the format with the DIFileTarget element. The DIFileTarget element must be placed within the DITransforms section of a data flow.

```
<DIFileTarget typeId="3" formatName="<FormatName>"
  filename="<FileName>">
```

Define the name of the target with a DIAttribute element.

```
<DIUIOptions>
<DIAttribute name="ui_display_name" value="<TargetName>" />
</DIUIOptions>
```

Specify a name for the input schema with the DIInputView element. By default, it is set to the format name. However, you can change it to any unique string, and use it in all downstream transforms.

```
<DIInputView name="<SchemaName>" />
```

Apply target-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="connection_port" value="no" />
  <DIAttribute name="file_location" value="local" />
  <DIAttribute name="isstreamdebugfile" value="no" />
  <DIAttribute name="loader_load_choice" value="replace" />
  ...
</DIAttributes>
```

End the target definition, with the closing tag of the DIFileTarget element.

```
</DIFileTarget>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording or location used in the Designer. For a complete description of each attribute, see the "Descriptions of objects" section in the *Designer Guide*.

DIAttribute name	Designer
connection_port	*Target File Editor › General › Make port*
file_location	*Target File Editor › Data File(s) › Location*
isstreamdebugfile	
loadler_load_choice	
name	*Target File Editor › General › Name*
root_dir	*Target File Editor › Data File(s) › Root directory*
transfer_custom	*Target File Editor › General › Custom transfer program*
validate_decimal_data	*Target File Editor › Default Format › Validate decimal data*

7.9.3.6 Database datastore

> ➜ Tip
>
> It's strongly recommended that you use the Designer to create and export your objects, and then use your application to adapt them as required. Many object options and parameters interact in complex ways, and do not always match the way they appear in the Designer.

Creating the object definition

Open the database datastore definition with the DIDatabaseDatastore element.

```
<DIDatabaseDatastore name="<datastore_name>" typeId="3">
```

Apply database datastore-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="DBLiveLoad" value="no" />
  <DIAttribute name="application_type" value="Custom" />
  <DIAttribute name="cdc_enabled" value="no" />
  <DIAttribute name="datastore_repotype" value="local" />
  ...
  <DIAttribute name="ds_configurations" hasNestedXMLTree="true">
```

Specify one or more datastore configurations inside the DIAttribute block with a DSConfigurations block. Only one configuration can be set as default.

```
    <DSConfigurations>
      <DSConfiguration default="true" name="<configuration_name>">
        <case_sensitive>no</case_sensitive>
        <database_type>Microsoft_SQL_Server</database_type>
        <loader_xact_size>1000</loader_xact_size>
        <locale_codepage>default</locale_codepage>
        <locale_language>default</locale_language>
        <locale_territory>default</locale_territory>
        <mssql_windows_authentication>
          no
        </mssql_windows_authentication>
        <password>;907A8897CEF453232929BD93946</password>
        <server_codepage>default</server_codepage>
        <sql_server_database>
          DS32_Source
        </sql_server_database>
        <sql_server_dataserver>
          testMachine
        </sql_server_dataserver>
        <sql_server_version>
          Microsoft SQL Server 2000
        </sql_server_version>
        <user>ods</user>
      </DSConfiguration>
    </DSConfigurations>
```

Close the DIAttribute block.

```
  </DIAttribute>
</DIAttributes>
```

End the database datastore definition with the closing tag of the DIDatabaseDatastore element.

```
</DIDatabaseDatastore>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording or location used in the Designer. The specific DIAttributes and XML elements available vary greatly depending on which type of database you are accessing. It's recommended that you create your datastore definitions using the Designer and then export them to XML.

For a complete description of each attribute, see the “Descriptions of objects” section in the *Designer Guide*.

7.9.3.7 Database table

> Tip
>
> It's strongly recommended that you use the Designer to create and export your objects, and then use your application to adapt them as required. Many object options and parameters interact in complex ways, and do not always match the way they appear in the Designer.

Creating the object definition

Open the database table definition with the DITable element.

```
<DITable name="<TableName>" owner="<Owner>"
  datastore="<DatastoreName>" database="<DatabaseName>"
  description="<Description>">
```

Apply table-specific properties with a DIProperties block. To define the table as a template table, set the value of the Loader_Is_Template_Table DIAttribute to `yes`.

```
<DIProperties>
  <DIAttribute name="Table_Type" value="TABLE" />
  <DIAttribute name="Estimated_Row_Count" value="50000" />
  <DIAttribute name="Loader_Is_Template_Table" value="no" />
  <DIAttribute name="db_alias_name" value="ODS" />
  ...
</DIProperties>
```

Specify column definitions with DIColumn elments. You can also define the content type for a column with the Content_Type attribute.

```
<DIColumn name="Cust_ID" datatype="VARCHAR" size="10"
  nullable="false" />
<DIColumn name="Cust_classf" datatype="VARCHAR" size="2"
  nullable="true" />
<DIColumn name="Address" datatype="VARCHAR" size="35"
  nullable="true" Content_Type="ADDRESS">
  <DIProperties>
    <DIAttribute name="Content_Type" value="ADDRESS" />
  </DIProperties>
</DIColumn>
```

Define the primary keys with a DIPrimaryKey block.

```
<DIPrimaryKey>
  <DIPrimaryKeyColumn name="Cust_ID" />
</DIPrimaryKey>
```

Define the unique table index with a DITableIndex element and specify the column name in a DIIndexColumn element.

```
<DITableIndex name="PK__ODS_CUSTOMER__7C8480A" unique="true">
  <DIIndexColumn name="Cust_ID" />
</DITableIndex>
```

End the database table definition with the closing tag of the DITable element.

```
</DITable>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording or location used in the Designer. For a complete description of each attribute, see the “Descriptions of objects” section in the *Designer Guide*.

7.9.3.7.1 To use as a source

To use a database table as a source within a data flow, invoke the table with the DIDatabaseTableSource element. The DIDatabaseTableSource element must be placed within the DITransforms section of a data flow.

```
<DIDatabaseTableSource typeId="22" datastoreName="<DatastoreName>"
  ownerName="<OwnerName>" tableName="<TableName>">
```

Specify a name for the output schema with the DIOutputView element.

```
<DIOutputView name="<SchemaName>" />
```

Apply source-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="array_fetch_size" value="1000" />
  <DIAttribute name="cache" value="yes" />
  <DIAttribute name="connection_port" value="no" />
  <DIAttribute name="enable_partitioning" value="no" />
  ...
</DIAttributes>
```

End the source definition, with the closing tag of the DIDatabaseTableSource element.

```
</DIDatabaseTableSource>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording or location used in the Designer. For a complete description of each attribute, see the “Descriptions of objects” section in the *Designer Guide*.

DIAttribute name	Designer
array_fetch_size	*Source Table Editor* > *Performance* > *Array fetch size*
cache	*Source Table Editor* > *Performance* > *Cache*
connection_port	*Source Table Editor* > *Make port*
enable_partitioning	
name	*Source Table Editor* > *Datastore name*
package_size	
reader_is_DB2CDC_table	
reader_overflow_file	
reader_template_table	
reader_use_overflow_file	
table_weight	*Source Table Editor* > *Performance* > *Join rank*

7.9.3.7.2 To use as a target

To use a database table as a target within a data flow, invoke the table with the DIDatabaseTableTarget element. The DIDatabaseTableTarget element must be placed within the DITransforms section of a data flow.

```
<DIDatabaseTableTarget typeId="11" bulkLoader="false"
  datastoreName="<DatastoreName>" ownerName="<OwnerName>"
  tableName="<TableName>">
```

Specify a name for the input schema with the DIInputView element.

```
<DIInputView name="<SchemaName>" />
```

Apply target-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="connection_port" value="no" />
        <DIAttribute name="loader_template_table" value="yes" />
  ...
  <DIAttribute name="ldr_configuration_enabled"
    value="yes" />
  <DIAttribute name="ldr_configurations"
    hasNestedXMLTree="true">
```

Specify the target configuration inside the DIAttribute block with a LDRConfigurations block.

```
    <LDRConfigurations>
      <LDRConfiguration database_type="Microsoft_SQL_Server"
```

```
        database_version="Microsoft SQL Server 2000">
        <auto_correct_using_merge>Yes</auto_correct_using_merge>
        <bulk_ldr_all_rows></bulk_ldr_all_rows>
        <bulk_ldr_max_errors></bulk_ldr_max_errors>
        <bulk_ldr_rows_per_commit></bulk_ldr_rows_per_commit>
        <enable_partitioning>no</enable_partitioning>
        <ignore_column_case>yes</ignore_column_case>
        <ignore_columns_null>No</ignore_columns_null>
        <ignore_columns_value></ignore_columns_value>
        <loader_auto_correct>no</loader_auto_correct>
        <loader_bulk_load>0</loader_bulk_load>
        ...
      </LDRConfiguration>
    </LDRConfigurations>
```

Close the DIAttribute block.

```
  </DIAttribute>
</DIAttributes>
```

End the target definition, with the closing tag of the DIDatabaseTableTarget element.

```
</DIDatabaseTableTarget>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording or location used in the Designer. The specific DIAttributes and XML elements available vary greatly depending on which type of database you are accessing. It's recommended that you create your database table targets using the Designer and then export them to XML.

For a complete description of each attribute, see the "Descriptions of objects" section in the *Designer Guide*.

DIAttribute name	Designer
connection_port	*Target Table Editor* › *Target* › *Make port*
name	*Target Table Editor* › *Target* › *Datastore name*
use_unicode_varchar	

7.9.3.8 Data Quality transforms

The options of Data Quality transforms have complex interactions and do not always match the way the transform configuration appears in the Designer. It's strongly recommended that you adapt exported versions of these transforms that have already been mostly configured how you want. By adapting existing objects, you can be more confident that the objects generated by your application will be free of problems, and function as you expect.

Because of the complexity of the Data Quality transforms, XSD files are provided for each transform. By default, the XSD files are installed to `LINK_DIR\Admin`.

> **Note**
>
> XML exported from the Designer may not validate correctly against the provided XSD files due to element ordering. However, any objects created by using the XSD files will operate correctly when imported to SAP Data Services.

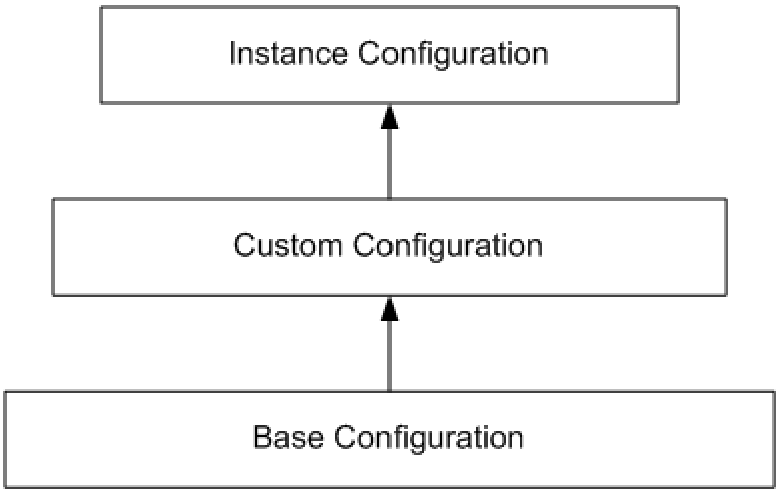
Related Information

Adapting objects [page 2314]

7.9.3.8.1 Hierarchy and inheritance

The ability to reuse transform configurations is a powerful feature of all Data Quality transforms available in SAP Data Services. To successfully use Data Quality transforms with the object creation XML toolkit, it's important to understand the rules of option inheritance and configuration reuse.

Data Quality transforms support three levels of configurations: base configuration, custom configuration, and instance configuration. Each level of configuration inherits the options and settings from the levels below it. However, settings defined explicitly at a higher level always take precedence over those inherited from a lower level.



In the Designer, the base configuration is simply represented in the form of the base transform. For example, the Global_Address_Cleanse transform. The base configuration carries the default settings for all options available for that transform. If you double-click any base transform, the configuration is opened in the Transform Configuration Editor. The Options tab lists all options set on the base configuration.

The custom configuration inherits the default settings from the base configuration, but also provides the ability to modify the values of any options at the custom level. In the Designer, the custom configuration level is represented by the available transform configurations. For example, for Global Address Cleanse, custom transform configurations such as Australia_AddressCleanse, Europe_AddressCleanse, and USA_AddressCleanse are available. For more information about transform configurations, see the “Data Flows, Transforms” section in the *Designer Guide*.

The instance configuration inherits default settings from the custom configuration, and provides the ability to further modify the values of any options. In the Designer, the instance configuration is represented by a transform instance within a data flow.

7.9.3.9 Query transform

🡆 Tip

It's strongly recommended that you use the Designer to create and export your objects, and then use your application to adapt them as required. Many object options and parameters interact in complex ways, and do not always match the way they appear in the Designer.

Creating the object definition

Open the query transform definition with the DIQuery element.

```
<DIQuery typeId="122">
```

Specify the name of the query transform and additional user interface options, such as the SQL where clause, in the DIUIOptions block.

```
<DIUIOptions>
  <DIAttribute name="ui_display_name" value="QryCDC" />
  <DIAttribute name="ui_where_text" value="<WhereClause>" />
</DIUIOptions>
```

Define the output schema with DIElement elements within a DISchema block. The input schema for each field is defined in the ui_mapping_text attribute, using the format **`<InputSchemaName>.<InputFieldName>`**.

```
<DISchema name="QryCDC">
  <DIElement name="CUST_ID" datatype="VARCHAR" size="10" key="true">
    <DIAttributes>
      <DIAttribute name="Description" value="" />
      <DIAttribute name="ui_mapping_text"
        value="ODS_CUSTOMER.CUST_ID" />
    </DIAttributes>
  </DIElement>

  <DIElement name="CUST_CLASSF" datatype="VARCHAR" size="2">
    <DIAttributes>
      <DIAttribute name="Description" value="" />
      <DIAttribute name="ui_mapping_text"
        value="ODS_CUSTOMER.CUST_CLASSF" />
    </DIAttributes>
  </DIElement>

  ...
</DISchema>
```

Define the expression for each output field with DIExpression elements within a DIProjection block. For exported XML, additional XML syntax for the output field may be present, but only the form used in the expr attribute is required.

```
<DISelect>
<DIProjection>
  <DIExpression isString="true" expr="ODS_CUSTOMER.CUST_ID">
    <COLUMN_REFERENCE qualifier1="ODS_CUSTOMER"
      column="CUST_ID" />
  </DIExpression>
  <DIExpression isString="true"
    expr="ODS_CUSTOMER.CUST_CLASSF" />
</DIProjection>
```

Specify the input schema in the DIFrom element.

```
<DIFrom>
  <DITableSpec name="ODS_CUSTOMER" />
</DIFrom>
```

Define the SQL where clause with a DIExpression element within a DIWhere block. For exported XML, additional XML syntax for the expression may be present, but only the form used in the expr attribute is required. The format is the same as it appears in the Designer.

```
<DIWhere>
  <DIExpression isString="true"
    expr="((ODS_CUSTOMER.CUST_TIMESTAMP &gt;=
      $GV_STARTTIME) AND$#xA; (ODS_CUSTOMER.CUST_TIMESTAMP
      &lt;= $GV_ENDTIME))">
    <!-- Additional XML syntax -->
  </DIExpression>
</DIWhere>
</DISelect>
```

Apply query-specific attributes with a DIAttribute block.

```
<DIAttributes>
  <DIAttribute name="distinct_run_as_separate_process"
    value="no" />
  <DIAttribute name="group_by_run_as_separate_process"
    value="no" />
  <DIAttribute name="join_run_as_separate_process"
    value="no" />
  <DIAttribute name="order_by_run_as_separate_process"
    value="no" />
  ...
</DIAttributes>
```

End the query transform definition with the closing tag of the DIQuery element.

```
</DIQuery>
```

Available DIAttribute names

The available DIAttribute names correspond to object parameters and properties, but do not necessarily match the wording or location used in the Designer. For a complete description of each attribute, see the “Descriptions of objects” section in the *Designer Guide*.

DIAttribute name	Designer
distinct_run_as_separate_process	*Query Editor ▸ Advanced ▸ Run DISTINCT as a separate process*
group_by_run_as_separate_process	*Query Editor ▸ Advanced ▸ Run GROUP BY as a separate process*
join_run_as_separate_process	*Query Editor ▸ Advanced ▸ Run JOIN as a separate process*
order_by_run_as_separate_process	*Query Editor ▸ Advanced ▸ Run ORDER BY as a separate process*
run_as_separate_process	

7.9.3.10 Parameters and variables

You can increase the flexibility and reusability of components generated by your application by using local and global variables when designing your jobs. For more information about including variables and parameters in the design of your jobs, see the “Variables and Parameters” section of the *Designer Guide*.

Global variables

Global variables can be accessed in XML within the DIScript element. For example, to assign a constant value to a global variable, you could use the DIAssignmentStep element:

```
<DIAssignmentStep typeId="12" variable="$GV_STARTTIME">
  <DIExpression isString="true"
    expression="'2001.01.01 00:00:000'" />
</DIAssignmentStep>
```

After the global variable is defined, you can use it elsewhere in script expressions used by your job. For example, you might insert the value of the variable into a table using a SQL query:

```
<DIFunctionCallStep typeId="23">
  <DIExpression isString="true"
    expr="sql('Target_DS', 'INSERT INTO TARGET.CDC_TIME
      VALUES ({$GV_STARTTIME})')" />
</DIFunctionCallStep>
```

For more information about using global variables in your jobs, see the “Variables and Parameters, Using global variables” section of the *Designer Guide*.

Substitution parameters

You can import one or more substitution parameter configurations to the repository with the Import_Repo_Object web service operation, as well as the Designer. In XML, substitution parameters are specified within the DISubVarStore and SVConfigurations elements.

```
<DISubVarStore typeId="103">
<SVConfigurations>
```

A default configuration can be specified with the DIAttribute element. This element is not required to import the substitution parameters.

```
<DIAttributes>
<DIAttribute name="SV_Config_Default"
  value="Configuration_1" />
</DIAttributes>
```

Specify a substitution parameter configuration with the SVConfiguration and SubVar elements. You can include more than one SVConfiguration block to specify additional substitution parameter configurations.

```
<SVConfiguration name="Configuration_1">
<SubVar name="ReportsAddressCleanse">1</SubVar>
<SubVar name="ReportsMatch">2</SubVar>
<SubVar name="USPSProviderLevel">3</SubVar>
<SubVar name="RefFilesAddressCleanse">4</SubVar>
...
</SVConfiguration>

</SVConfigurations>
</DISubVarStore>
```

After importing the substitution parameter configurations to the repository, they can be used normally in your data flows. For more information about using substitution parameters, see the “Variables and Parameters, Substitution parameters” section of the *Designer Guide*.

7.9.3.11 Basic example

This example assumes that you have a simple job named myTestJob that calls a data flow named myTestDataflow. The data flow contains a flat-file source (mySource), a Query transform (FormatFields), and a flat-file target (myTarget).

Additionally, this example assumes that all objects for the job have been exported to a single XML file.

Example

All object definitions are enclosed in the DataIntegratorExport element.

```
<?xml version="1.0" encoding="UTF-8" ?>
<DataIntegratorExport repositoryVersion="12.2.0.0000"
  productVersion="12.2.0.0">
```

The NameDate flat-file format definition is contained in the DIFlatFileDatastore element. Field names, types, and sizes are defined in the DISchema block.

```
<DIFlatFileDatastore name="NameDate" typeId="3">

<DISchema>
  <DIElement name="FirstName" datatype="VARCHAR" size="7">
  </DIElement>
  <DIElement name="LastName" datatype="VARCHAR" size="7">
  </DIElement>
  <DIElement name="DateOfBirth" datatype="VARCHAR" size="10">
  </DIElement>
</DISchema>
```

Various attributes, such as file name, directory, file type, column delimiter, and locale settings are specified in the DIAttributes block, and the flat-file format definition is closed.

```
<DIAttributes>
<DIAttribute name="abap_file_format" value="no" />
<DIAttribute name="blank_pad" value="leading" />
<DIAttribute name="cache" value="yes" />
<DIAttribute name="column_delimiter" value="," />
<DIAttribute name="column_width" value="1" />
<DIAttribute name="column_width1" value="7" />
<DIAttribute name="column_width2" value="7" />
<DIAttribute name="column_width3" value="10" />
<DIAttribute name="date_format" value="yyyy.mm.dd" />
<DIAttribute name="datetime_format"
  value="yyyy.mm.dd hh24:mi:ss" />
<DIAttribute name="file_format" value="ascii" />
<DIAttribute name="file_location" value="local" />
<DIAttribute name="file_name" value="NameDate.txt" />
<DIAttribute name="file_type" value="delimited_file" />
<DIAttribute name="locale_codepage" value="&lt;default&gt;" />
<DIAttribute name="locale_language" value="&lt;default&gt;" />
<DIAttribute name="locale_territory"
  value="&lt;default&gt;" />
<DIAttribute name="name" value="NameDate" />
<DIAttribute name="reader_capture_data_conversion_errors"
  value="no" />
<DIAttribute name="reader_capture_row_format_errors"
  value="yes" />
<DIAttribute name="reader_error_file_name" value="" />
<DIAttribute name="reader_error_file_root_dir" value="" />
<DIAttribute name="reader_log_data_conversion_warnings"
  value="yes" />
<DIAttribute name="reader_log_row_format_warnings"
  value="yes" />
<DIAttribute name="reader_maximum_warnings_to_log"
  value="-99" />
<DIAttribute name="reader_write_error_rows_to_file"
  value="no" />
<DIAttribute name="root_dir"
  value="C:\Data Services\Tutorial Files" />
<DIAttribute name="row_delimiter" value="\n" />
<DIAttribute name="skip_row_header" value="yes" />
<DIAttribute name="table_weight" value="0" />
<DIAttribute name="time_format" value="hh24:mi:ss" />
<DIAttribute name="transfer_custom" value="no" />
<DIAttribute name="use_root_dir" value="no" />
<DIAttribute name="write_bom" value="no" />
</DIAttributes>

</DIFlatFileDatastore>
```

The myTestJob job definition is contained in the DIJob element. The job contains a single step, calling the myTestdata flow data flow. Again, attributes related to tracing, statistics, and so on are found in the DIAttributes block.

```
<DIJob name="myTestJob" typeId="2">

<DISteps>
<DICallStep typeId="1" calledObjectType="Dataflow"
  name="myTestDataflow" >
</DICallStep>
</DISteps>

<DIAttributes>
<DIAttribute name="job_checkpoint_enabled" value="no" />
<DIAttribute name="job_collect_statistics" value="no" />
<DIAttribute name="job_collect_statistics_monitor"
  value="no" />
<DIAttribute name="job_enable_assemblers" value="yes" />
<DIAttribute name="job_enable_audit" value="yes" />
<DIAttribute name="job_enable_dataquality" value="yes" />
<DIAttribute name="job_export_repo" value="no" />
<DIAttribute name="job_export_reports" value="no" />
<DIAttribute name="job_isrecoverable" value="no" />
<DIAttribute name="job_mode" value="Multi-Process" />
<DIAttribute name="job_monitor_sample_rate" value="1000" />
<DIAttribute name="job_name" value="myTestJob" />
<DIAttribute name="job_print_version" value="no" />
<DIAttribute name="job_testmode_enabled" value="no" />
<DIAttribute name="job_trace_abapquery" value="no" />
<DIAttribute name="job_trace_all" value="no" />
<DIAttribute name="job_trace_ascomm" value="no" />
<DIAttribute name="job_trace_assemblers" value="no" />
<DIAttribute name="job_trace_audit" value="no" />
<DIAttribute name="job_trace_dataflow" value="yes" />
<DIAttribute name="job_trace_idoc_file" value="no" />
<DIAttribute name="job_trace_memory_loader" value="no" />
<DIAttribute name="job_trace_memory_reader" value="no" />
<DIAttribute name="job_trace_optimized_dataflow"
  value="no" />
<DIAttribute name="job_trace_parallel_execution"
  value="no" />
<DIAttribute name="job_trace_rfc_function" value="no" />
<DIAttribute name="job_trace_row" value="no" />
<DIAttribute name="job_trace_script" value="no" />
<DIAttribute name="job_trace_session" value="yes" />
<DIAttribute name="job_trace_sql_only" value="no" />
<DIAttribute name="job_trace_sqlfunctions" value="no" />
<DIAttribute name="job_trace_sqlloaders" value="no" />
<DIAttribute name="job_trace_sqlreaders" value="no" />
<DIAttribute name="job_trace_sqltransforms" value="no" />
<DIAttribute name="job_trace_stored_procedure" value="no" />
<DIAttribute name="job_trace_table" value="no" />
<DIAttribute name="job_trace_table_reader" value="no" />
<DIAttribute name="job_trace_transform" value="no" />
<DIAttribute name="job_trace_userfunction" value="no" />
<DIAttribute name="job_trace_usertransform" value="no" />
<DIAttribute name="job_trace_workflow" value="yes" />
<DIAttribute name="job_type" value="batch" />
<DIAttribute name="job_use_statistics" value="yes" />
<DIAttribute name="locale_codepage"
  value="&lt;default&gt;" />
<DIAttribute name="locale_language"
  value="&lt;default&gt;" />
<DIAttribute name="locale_territory"
  value="&lt;default&gt;" />
</DIAttributes>
```

```
</DIJob>
```

The myTestdata flow data flow definition is contained in the DIDataflow element.

```
<DIDataflow name="myTestDataflow" typeId="1">
```

Transforms are invoked within the DITransforms element. Because this data flow has three transforms, source, Query, and target, there are three corresponding sections in the DITransforms element.

```
<DITransforms>
```

The file-format source definition is contained in the DIFileSource element. The name of the output schema is specified with the DIOutputView element. By default, the schema is given the same name as the file format. However, you can change it to any unique name, provided you use the same name in later transforms.

```
<DIFileSource typeId="33" formatName="NameDate"
  filename="NameDate.txt">

<DIUIOptions>
<DIAttribute name="ui_display_name" value="mySource" />
</DIUIOptions>

<DIOutputView name="NameDate" />

<DIAttributes>
<DIAttribute name="adaptable" value="no" />
<DIAttribute name="cache" value="yes" />
<DIAttribute name="connection_port" value="no" />
<DIAttribute name="file_location" value="local" />
<DIAttribute name="name" value="NameDate" />
<DIAttribute name="reader_filename_col"
  value="DI_FILENAME" />
<DIAttribute name="reader_filename_col_size" value="100" />
<DIAttribute name="reader_filename_only" value="no" />
<DIAttribute name="reader_include_filename" value="no" />
<DIAttribute name="reader_maximum_warnings_to_log"
  value="-99" />
<DIAttribute name="root_dir"
  value="C:\Data Services\Tutorial Files" />
<DIAttribute name="table_weight" value="0" />
</DIAttributes>

</DIFileSource>
```

The FormatFields Query transform definition is contained in the DIQuery element. The output schema name is specified with the DISchema element.

```
<DIQuery typeId="122" >

<DIUIOptions>
<DIAttribute name="ui_display_name" value="FormatFields" />
</DIUIOptions>

<DISchema name="FormatFields">
```

Each output field is defined with DIElement and DIAttributes elements. The ui_mapping_text attribute for each output field is required by the Designer. For the Full Name output field, the expression shows the concatenation of two input fields (FirstName and LastName). The additional encoded text is used to maintain formatting within the Designer and is optional.

```
<DIElement name="FirstName" datatype="VARCHAR" size="7">
<DIAttributes>
<DIAttribute name="Description" value="" />
<DIAttribute name="ui_mapping_text"
  value="NameDate.FirstName" />
</DIAttributes>
</DIElement>

<DIElement name="LastName" datatype="VARCHAR" size="7">
<DIAttributes>
<DIAttribute name="Description" value="" />
<DIAttribute name="ui_mapping_text"
  value="NameDate.LastName" />
</DIAttributes>
</DIElement>

<DIElement name="DateOfBirth" datatype="VARCHAR" size="10">
<DIAttributes>
<DIAttribute name="Description" value="" />
<DIAttribute name="ui_mapping_text"
  value="NameDate.DateOfBirth" />
</DIAttributes>
</DIElement>

<DIElement name="Full Name" datatype="VARCHAR" size="20">
<DIAttributes>
<DIAttribute name="Description" value="" />
<DIAttribute name="ui_mapping_text"
  value="NameDate.FirstName ||
    NameDate.LastName&#xD;&#xA;&#xD;&#xA;" />
</DIAttributes>
</DIElement>

</DISchema>
```

The SQL select projection syntax is contained in the DISelect and DIProjection elements, and varies depending on how the Query transform is configured.

```
<DISelect>
<DIProjection>
```

Each output field is defined in a DIExpression element. The expr attribute contains the actual ATL expression as displayed in the Designer. Note that each field contains an additional XML representation of the expression. This additional representation is optional, and not required for correct operation.

```
<DIExpression isString="true" expr="NameDate.FirstName">
<COLUMN_REFERENCE qualifier1="NameDate" column="FirstName" />
</DIExpression>

<DIExpression isString="true" expr="NameDate.LastName">
<COLUMN_REFERENCE qualifier1="NameDate" column="LastName" />
</DIExpression>

<DIExpression isString="true" expr="NameDate.DateOfBirth">
<COLUMN_REFERENCE qualifier1="NameDate"
  column="DateOfBirth" />
</DIExpression>

<DIExpression isString="true" expr="(NameDate.FirstName ||
  NameDate.LastName)">
<CONCAT>
```

```
<COLUMN_REFERENCE qualifier1="NameDate" column="FirstName" />
<COLUMN_REFERENCE qualifier1="NameDate" column="LastName" />
</CONCAT>
</DIExpression>

</DIProjection>
```

The input schema for the Query is defined using the DIFrom element. Attributes for the Query are specified in a DIAttributes block.

```
<DIFrom>
<DITableSpec name="NameDate" />
</DIFrom>
</DISelect>

<DIAttributes>
<DIAttribute name="distinct_run_as_separate_process"
  value="no" />
<DIAttribute name="group_by_run_as_separate_process"
  value="no" />
<DIAttribute name="join_run_as_separate_process"
  value="no" />
<DIAttribute name="order_by_run_as_separate_process"
  value="no" />
<DIAttribute name="run_as_separate_process" value="no" />
</DIAttributes>

</DIQuery>
```

The file-format target is defined within the DIFileTarget element

```
<DIFileTarget typeId="3" formatName="NameDate"
  filename="NameDate_out.txt">

<DIUIOptions>
<DIAttribute name="ui_display_name" value="myTarget" />
</DIUIOptions>
```

The input schema for the target is specified using the DIInputView element, and attributes for the target are specified using a DIAttributes block.

```
<DIInputView name="FormatFields" />

<DIAttributes>
<DIAttribute name="connection_port" value="no" />
<DIAttribute name="file_location" value="local" />
<DIAttribute name="isstreamdebugfile" value="no" />
<DIAttribute name="loader_load_choice" value="replace" />
<DIAttribute name="name" value="NameDate" />
<DIAttribute name="root_dir" value="D:\temp" />
<DIAttribute name="validate_decimal_data" value="yes" />
</DIAttributes>

</DIFileTarget>

</DITransforms>
```

Attributes for the myTestdata flow data flow are specified in a DIAttributes block, and the file is finished by closing the DataIntegratorExport element.

```
<DIAttributes>
<DIAttribute name="Cache_type" value="pageable_cache" />
<DIAttribute name="Parallelism_degree" value="0" />
```

```
<DIAttribute name="allows_both_input_and_output"
  value="yes" />
<DIAttribute name="run_once" value="no" />
<DIAttribute name="use_dataflow_links" value="no" />
<DIAttribute name="use_datastore_links" value="yes" />
<DIAttribute name="validation_xform_exists" value="no" />
<DIAttribute name="validation_xform_stats" value="no" />
</DIAttributes>

</DIDataflow>

</DataIntegratorExport>
```

7.10 Job launcher execution commands

The job launcher, exported as part of a job's execution commands, includes a specific command line option for server groups. You can use this option to change the job servers in a server group.

For complete information about the job launcher, see the *Management Console Guide*.

The following table lists job launcher flags and their values.

Flag	Value
`-w`	The job launcher starts the job(s) and then waits before passing back the job status. If -w is not specified, the launcher exits immediately after starting a job.
`-t`	The time, in milliseconds, that the Job Server waits before checking a job's status. This is a companion argument for `-w`.
`-s`	Status or return code. 0 indicates successful completion, non-zero indicates an error condition. Combine `-w`, `-t`, and `-s` to execute the job, wait for completion, and return the status.
`-C`	Name of the engine command file (path to a file which contains the Command line arguments to be sent to the engine).
`-v`	Prints `AL_RWJobLauncher` version number.
`-S`	Lists the server group and Job Servers it contains using the following syntax: `"SvrGroup-Name;JobSvr1Name:JobSvr1Host:JobSvr1Port;JobSvr2Name:JobSvr2Host:JobSvr2Port";` For example: `"SG_DEV;JS1:HPSVR1:3500;JS2:WINSVR4:3505";`
`-R`	The location and name of the password file. Replaces the hard-coded repository connection values for `-S, -N, -U, -P`.

There are two arguments that do not use flags:

- `inet` address—The host name and port number of the Job Server. The string must be in quotes. For example:

  ```
  "inet:HPSVR1:3500"
  ```

  If you use a server group, inet addresses are automatically rewritten using the `-S` flag arguments. On execution, the first Job Server in the group checks with the others and the Job Server with the lightest load executes the job.
- `server log path`—The fully qualified path to the location of the log files. The server log path must be in quotes. The server log path argument does not appear on an exported batch job launch command file. It appears only when the software generates a file for an active job schedule and stores it in the following directory: **`<DS_COMMON_DIR>`**`/Log/`**`<JobServerName>`**`/`**`<RepositoryName>`**`/`**`<JobInstanceName>`**
  You cannot manually edit server log paths.

7.11 Legacy adapter information

7.11.1 Legacy adapter for external web services

> ⚠ Caution
>
> This section is provided for legacy reference only. For improved performance in new web service data flows, use the native web service datastore type.

You can add functionality to SAP Data Services to invoke web services in external applications from data flows. This functionality requires configuring the software's built-in Web Services Adapter. The Web Services Adapter provides support for locating and importing metadata for a web services server as well as invoking web service operations.

The Web Services Adapter works by sending a request and waiting until it receives a reply from a web services server.

For example, you might create a web services server as a front-end to a legacy application. You could call the web services server daily from a data flow to access inventory and update an inventory data mart.

The interaction between the Web Services Adapter and an external web service has these parts:

- Creating an adapter datastore that identifies the WSDL, which describes the web services server.
- Importing metadata to extract the information form the WSDL needed to access the web service server.
- Creating a data flow that uses the imported function call to call the web services server.

Related Information

To access a web service using the Designer [page 2258]

7.11.1.1 Legacy adapter installation

The Web Services Adapter is part of each Job Server installation. The installer automatically configures an adapter instance in the Administrator, which is the only adapter instance that SAP Data Services requires to configure a web services client. You do not need to configure adapter operations. The software automatically configures the Web Services Adapter with *Autostart* set to `FALSE` so that it does not consume resources when you do not use Web services. However, you can invoke adapters set with *Autostart* disabled, when needed. You do not need to edit the adapter instance that the software provides for the Web Services Adapter.

The installer allows you to configure a Job Server to manage adapters by presenting a list of Job Servers to you during installation. To view any adapter instance in the Administrator, select *Adapter Instance* **<Job Server>**.

The software creates the following values for an adapter instance.

Field name	Value automatically inserted
Adapter Instance Name	WebService
Access Sever host	(not required) Blank
Access Server port	(not required) Blank
Adapter Retry count	(default value) 0
Adapter Retry Interval	(default value) 3000
ClassPath	JAR files required in the classpath to start the Java process: **<LINK_DIR>**/lib/acta_adapter_sdk.jar **<LINK_DIR>**/lib/acta_broker_client.jar **<LINK_DIR>**/lib/acta_tool.jar **<LINK_DIR>**/ext/lib/xerces.jar **<LINK_DIR>**/lib/acta_webservice_adapter.jar **<LINK_DIR>**/ext/lib/qname.jar **<LINK_DIR>**/ext/lib/axis.jar **<LINK_DIR>**/ext/lib/commons-logging.jar **<LINK_DIR>**/ext/lib/commons-discovery.jar **<LINK_DIR>**/ext/lib/wsdl4j.jar **<LINK_DIR>**/ext/lib/saaj.jar **<LINK_DIR>**/ext/lib/jaxrpc.jar
AutoStart	TRUE
Trace Mode	FALSE (If set to TRUE, the adapter writes trace messages to the `WebService_trace.txt` file in the **<DS_COMMON_DIR>**/`adapters/log` directory)
Additional Java launcher options	(default values) -Xms64m -Xmx256m

Field name	Value automatically inserted
Adapter type name	(Read-only) Name of adapter used to create this instance
Adapter version	(Read-only) Version of adapter used to create this instance
Adapter class	(Read-only) Name that identifies the adapter entry point

7.11.1.2 Legacy adapter configuration

To configure access to a specific web service, use the Designer. In the Designer's Datastore Editor window, specify the datastore as an adapter datastore, select the Job Server that is managing the Web Services Adapter, and select the Web Services Adapter. Data Services provides access to web services as stream-oriented function calls, which it configures when you import metadata.

When you configure an adapter datastore, in addition to the normal adapter settings, specify the URL of the web services server for a data flow to access. It must be the same URL that accepts a web service connection and returns the WSDL.

The adapter connects to the web services server using the URL to locate the definitions of published services.

7.11.1.2.1 To access a web service

1. Create an adapter datastore:
   a) Use the Web Services Adapter instance that SAP Data Services automatically creates during installation.
   b) In the datastore editor under Adapter Options, configure the following parameters:
      - *URL of the Web Service* — Enter the URL of the web services server. This URL must accept a web service connection and return the WSDL. This information is required for data flow access.
      - *XML Recursion Level* — Enter the number of passes the software should run through the XSD to resolve names. The default is 0.
      - *Keystore Path* — If the web services server uses an SSL connection, specify the location of the keystore used to establish the connection. When unsure, contact your network administrator.
      - *User Name* — Enter your user name for HTTP basic authentication.
      - *Password* — Enter your password for HTTP basic authentication.

      > Note
      >
      > To obtain user name and password information, contact your web services provider.

   c) Click *OK*.
2. Import metadata:
   a) From the object library, double-click a Web Services Adapter datastore.

      The Designer calls the adapter. The adapter calls the web services server at the indicated WSDL URL and obtains a list of published services and ports.
   b) Expand the ports to see the published operations available for import.

The list reflects the name and description of operations currently published by the configured web service.

c) Right-click an operation and select *Import*.

The software imports web service operations as function calls and lists them under the Web Services Adapter datastore in the object library. Each function call includes a definition for both the input and output messages required for communication with a web service operation. The adapter extracts the details about the request and reply messages and generates XML Schema files that describe the messages.

3. From the Designer, add a web service function call to a job.

   As a web services client, the software calls a web services server twice:

   - During design time to import metadata for the functions and data types that a particular web service supports.
   - During run time to call the web service and invoke this functionality.

7.11.1.2.2 To add web service calls to a job

Once an adapter datastore is created and metadata is imported, use the following procedure to add a function call to an SAP Data Services job.

1. Open the Designer.
2. Create a Web Services Adapter datastore.
3. Import operation metadata from an external web service.
4. Add a query to your job.
5. Open the query editor, right-click the target schema and select *New Function call*.

   The Function Editor opens listing the operation metadata that you imported under its datastore name.
6. Select a datastore to view the metadata that you want to add to your job.
7. Select the metadata name and click *Next*.
8. Map the input schema to the output schema.

   > Note
   >
   > If you want to nest data in the target schema, use this first query to place the schema in your job and additional queries to perform the nesting. The Function Editor does not allow complex schema configuration.
9. Click *OK*.

   The imported schema appears in the query.
10. Configure the remainder of your job by supplying input to the function call and extracting the response information obtained from the web service.

7.11.1.3 Configuring SSL with the legacy adapter

With Secure Socket Layer (SSL), the web services adapter can use secure transport over TCP/IP networks.

The overall process is:

- Generate certificates and keystores for both the server and client.
- Configure the web server.
- Configure the SAP Data Services web services adapter.

7.11.1.3.1 To generate certificates and keystores

1. Generate the server keystore.
2. Export the certificate from the server keystore to a file and get it signed by an authorized Certificate Authority.
3. Generate the client keystore.
4. Export the certificate from the client keystore to a file and get it signed by an authorized Certificate Authority.
5. Import the client's certificate into the server's keystore.
6. Import the server's certificate into client's keystore.

7.11.1.3.2 To configure Tomcat and the legacy adapter

1. Uncomment the following entry from the `server-di.xml` file in the `TOMCAT_HOME/conf` directory.

```
<Connector port="8443"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" disableUploadTimeout="true"
acceptCount="100" debug="1" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />
```

2. Inside the `Connector` tag, add/update the value of the `keystoreFile` and `keystorePass` parameters. The `keystoreFile` parameter should contain the *.keystore file path created when generating the keystore. The `keystorePass` parameter should contain the password used to create the keystore when generating it.
3. You might also need to modify the `wrapper.properties` file in the `LINK_DIR\ext\webserver\conf` directory. In the section `Defining the classpath`, ensure the following third-party libraries are in the path:

```
wrapper.class_path=$(ACTAHOME)\ext\lib\jnet.jar
wrapper.class_path=$(ACTAHOME)\ext\lib\jsse.jar
wrapper.class_path=$(ACTAHOME)\ext\lib\jcert.jar
```

4. After completing the configuration changes, restart the web server. If the web server starts successfully, then you should be able to access any web application supported by Tomcat via SSL.
5. In the Designer, configure the web services adapter by opening the adapter datastore, click *Advanced*, and enter the client keystore path.

7.11.1.4 Legacy adapter error messages

In addition to the error logs, the following list identifies web services client error messages and their descriptions:

- Web services client is unable to create a SOAP request to send to a server. Error = <adapter-generated exception message>
  The Web Services Adapter returns this error message if the XML message passed from SAP Data Services as a Web Services Adapter function call could not be packaged into a SOAP Envelope. Processing stops before a call to a web services server is made.

  To find extended error information, see the Web Services Adapter trace log file (▶ *Adapter Instances* ▶ **<JobServerName>** ▶). To use extended diagnostics, use debug tracing in the `webadmin.log` file.
- Web services client is unable to invoke a web services server. Error = <adapter-generated exception message>
  The client returns this error message if the Web Services Adapter cannot call the web services server. It indicates that the adapter has successfully packaged the XML message passed from the software into a SOAP Envelope. However, the call to the web services server is not going through. In most cases, this will be an error in locating the service, not a case of the service refusing a request. If a web service operation is refusing a request, it will return a fault message.

  To find extended error information, see the Web Services Adapter trace log file (▶ *Adapter Instances* ▶ **<JobServerName>** ▶). To use extended diagnostics, use debug tracing in the `webadmin.log` file.
- Web services client called a web services server. The server returned the following fault message: <server-generated error message>
  The client returns this error message if the web services server is called and returns a fault message indicating the call failed. The adapter has successfully packaged the XML message passed from the software into a SOAP Envelope, called the web services server, and the server received the call.

  To find extended error information, see the Web Services Adapter trace log file (▶ *Adapter Instances* ▶ **<JobServerName>** ▶) and perhaps the server itself if it maintains diagnostics. To use extended diagnostics, use debug tracing in the `webadmin.log` file.
- Web services client called a web services server and received a reply that cannot be interpreted. Error = < adapter-generated exception message >
  The client returns this error message if the SOAP Envelope returned from the web services server cannot be unpacked to extract the XML message to be returned to the job.

  To find extended error information, see the Web Services Adapter trace log file (▶ *Adapter Instances* ▶ **<JobServerName>** ▶). To use extended diagnostics, use debug tracing in the `webadmin.log` file.

8 Supplement for J.D. Edwards

8.1 Overview

The SAP Data Services J.D. Edwards (JDE) interface is a license-controlled feature. With this interface, you can use the Designer to:

- View modules and tables in a database instance under a specified J.D. Edwards environment
- Import metadata for tables into SAP Data Services
- Create batch data flows that use J.D. Edwards tables as sources

Related Information

System requirements [page 2353]
Datastores [page 2354]

8.2 System requirements

The J.D. Edwards interface allows you to connect SAP Data Services with two J.D. Edwards applications:

- World
- OneWorld

Both of these applications operate on an underlying database. Install the drivers you need to connect J.D. Edwards with SAP Data Services on the same computers on which you install the Designer and Job Server components.

8.2.1 World

The J.D. Edwards interface supports World version A7.3 and later. SAP Data Services only supports this application when World uses an underlying DB2 database on an AS/400 computer. Use either an ODBC datastore connection (using IBM's iSeries Access driver) or a Detail_DB2 datastore connection.

Related Information

Defining a J.D. Edwards World datastore [page 2354]

8.2.2 OneWorld

The J.D. Edwards interface supports OneWorld version B7.3 and compatible versions, including Xe. SAP Data Services supports this application if OneWorld uses one of three underlying databases:

- DB2 for AS/400 (use an ODBC datastore connection using IBM's iSeries Access)
- Microsoft SQL Server
- Oracle

J.D. Edwards OneWorld environment data sources include system-control data sources and business data sources. Your J.D. Edwards system-control data sources and your business data sources must reside on the same database instance. If a system control table is not located in the same database instance, you cannot create a datastore for that J.D. Edwards application.

Related Information

Defining a J.D. Edwards OneWorld datastore [page 2356]

8.3 Datastores

SAP Data Services uses datastore connections to link with other applications or databases. In a design environment, you use datastores to browse, search, or import metadata that represents external tables, files, messages, and other database objects. When running jobs, the software uses datastore information to move data between source and target databases and applications.

After defining a J.D. Edwards datastore in SAP Data Services, you can browse application modules, tables, table descriptions, and column descriptions.

You must define J.D. Edwards datastore connections with accurate J.D. Edwards information to ensure the accessibility of the tables.

8.3.1 Defining a J.D. Edwards World datastore

You can use two methods to access the J.D. Edwards World application:

- ODBC driver
- Mainframe interfaces

Both options access a DB2 for AS/400 database.

8.3.1.1 ODBC driver

You can access the J.D. Edwards World application using an ODBC driver called iSeries Access. Before creating a datastore using this ODBC driver, you must install and configure the driver on the same computers on which you installed the Designer and Job Server components. Use the ODBC Administration utility to install and configure the driver. In the driver's library list, be sure to list all the libraries needed in your particular J.D. Edwards environment.

8.3.1.2 Mainframe interfaces

You can access the J.D. Edwards World application using Mainframe interfaces. Refer to the pertinent documentation for installation and setup instructions.

8.3.1.2.1 To define a J.D. Edwards World datastore

1. In the object library of the Designer, go to the *Datastores* tab.
2. Right-click inside the object library window and select *New*.
3. In the Datastore Editor window, enter a name for this datastore (`DS_JDEWorld`, for example).
4. In the *Application type* list, select *JDE World*.
5. In the *Database Type* list, select *ODBC*.
6. Enter the connection information.
7. Click the *JDE World Properties* tab and enter the required information. SAP Data Services handles J.D. Edwards data in a logical grouping of libraries.

Option	Description
Environment	Enter the J.D. Edwards application environment name.
Local library	Enter the name of the library where SAP Data Services can find the J.D. Edwards system control table F0005.
Data dictionary library	Enter the name of the library where SAP Data Services can find the J.D. Edwards data dictionary table F9201.
Security library	Enter the name of the library where SAP Data Services can find J.D. Edwards security tables, such as F0094.

8. Click *OK*.

   The J.D. Edwards datastore appears in the object library.

Related Information

Reference Guide: Objects, Database datastores [page 868]

8.3.2 Defining a J.D. Edwards OneWorld datastore

A J.D. Edwards environment uses a logical construct called a data source to associate the J.D. Edwards application layer data with the database layer data (for example, Microsoft SQL Server). J.D. Edwards uses some data sources to associate system-control data and uses other data sources to associate business data.

The distributed nature of J.D. Edwards architecture allows an instance of J.D Edwards to have data sources on multiple database servers. For example, a finance data source can be on a Microsoft SQL Server while the system-control data source is on Oracle. Because a SAP Data Services datastore can only be associated with one database server, you must follow these rules when defining a J.D. Edwards OneWorld datastore connection in SAP Data Services:

- The data sources within a J.D. Edwards environment that you want to access must be in the same Microsoft SQL Server, DB2 for AS/400, or Oracle instance.
- You must specify required system control data sources, and in some cases, you must enter the owner ID of the data source. For Microsoft SQL Server, you must also enter the database name for each, and for DB2 for AS/400, you must enter the library for each.
- If using Microsoft SQL Server, you must configure a new datastore for each business data source on a unique database even if all databases are on the same Microsoft SQL Server. For example, if you want to extract Human Resource and Finance information from separate databases on the same Microsoft SQL Server instance, you must create two different datastores.

8.3.2.1 To define a J.D. Edwards OneWorld datastore

1. In the object library of the Designer, go to the *Datastores* tab.
2. Right-click inside the object library window and select *New*.
3. In the Datastore Editor window, enter a name for this datastore (`DS_JDEOneWorld`, for example).
4. In the *Application Type* list, select *JDE_One_World*.
5. In the *Database Type* list, select the database on which your J.D. Edwards application runs.

If you choose:	Do this:
Microsoft SQL Server	Enter the database-related information including the user name and password for the database. In the *Database server name* box, enter the name of the SQL Server instance. In the *Database name* box, enter the name of the database containing your business data, such as a finance database. If you have more than one database of business data create a separate J.D. Edwards datastore like the first except enter a different database name here.

If you choose:	Do this:
	Note With Microsoft SQL Server as your database, the Job Server must be installed on Windows NT or 2000.
Oracle	Enter the information required to connect to the Oracle database.

6. Click the *JDE OneWorld Properties* tab and enter the required information: the environment name and the databases to which the data sources point.

   Some J.D. Edwards data sources contain system-control information. SAP Data Services needs this system-control information to properly translate J.D. Edwards data. Like all data sources, these system-control data sources can be anywhere on the Microsoft SQL Server, DB2 for AS/400, or Oracle instance. Enter the database names for each data source.

Option	Description
Environment	Enter the J.D. Edwards application environment name.
System data source	Enter the name of the database where the tables F986101, F98611, and F00941 are located. This option is available for DB2 and Microsoft SQL Server databases.
System data source owner	Enter the owner ID for the system data source. This option is available for Microsoft SQL Server and Oracle databases.
Object librarian data source	Enter the name of the database where the tables F9860 and F9861 are located. This option is available for DB2 and Microsoft SQL Server databases.
Local data source	Enter the name of the database where the table F0005 is located. This option is available for DB2 and Microsoft SQL Server databases.
Data dictionary data source	Enter the name of the database where the table F9203 is located. This option is available for DB2 and Microsoft SQL Server databases.

7. Click *OK*.
   The J.D. Edwards OneWorld datastore appears in the object library.

8.3.3 Browsing and importing metadata

After creating a J.D. Edwards datastore, you can browse and import the metadata from the connected application.

8.3.3.1 To browse and import the metadata

1. View the modules in the J.D. Edwards application. You can:
   - ○ Right-click the datastore name and select *Open*
   - ○ Double-click the datastore name

   The workspace shows the list of application modules.
2. View the tables in a particular module. You can:
   - ○ Expand the module tree
   - ○ Double-click a module (folder)
3. To import table data, right-click a table name and select *Import*.

   > Note
   >
   > When you import a table into SAP Data Services via the J.D. Edwards interface, the software does not preserve the table hierarchy. All tables are listed at the same level.

You can also import tables using the Designer *Import by Name* and *Search* features.

8.3.4 Extracting data from J.D. Edwards systems

When extracting data from either J.D. Edwards application, SAP Data Services processes and converts data types appropriately. For example, the software recognizes null values for date columns. J.D. Edwards, on the other hand, does not support null values. Instead, J.D. Edwards stores dates as numeric values. When date data is not present, J.D. Edwards stores the number 0. Therefore, the software automatically translates a value of zero for a date to NULL. If you expect a column to have zeros in it, do not use the date type in the primary key.

SAP Data Services automatically translates most data types. In some cases, you must translate output explicitly.

Related Information

Data types [page 2358]

8.4 Reference information

8.4.1 Data types

SAP Data Services uses unique processing to translate some J.D. Edwards data types:

- Translating decimals

- Translating dates
- Translating time

8.4.1.1 Translating decimals

J.D. Edwards translates decimal values using an additional piece of information from the underlying database: the decimal separator shift integer. This value indicates the number of digits to move left from the end of the value number. For example, to represent a number like 1.23, J.D. Edwards stores the value as "123" with a decimal separator shift of "2" (starting at the end of the number, the decimal shifts two places to the left).

To process J.D. Edwards decimal values, the software extracts the numeric value stored in the underlying database and then applies the decimal separator shift to determine where the decimal belongs. The software applies the translation when a decimal column appears in a SQL statement against the database.

SAP Data Services interprets J.D. Edwards currency data types as decimals.

The software does not automatically translate decimal data types in three cases:

- SQL transform
- sql function
- pushdown_sql function

In these cases, translate decimals explicitly. To translate decimals explicitly, you must know the decimal shift value for your selected column. Use the shift number to determine the denominator for decimal translation. For example, if column `COL26` in table `JDETAB` is a JDE decimal column and it has a shift value of `2`, you would manually modify the SQL statement to say something like `SELECT COL26/100 FROM JDETAB`. The denominator in this statement has two zeros following the `1`. If the shift value was `3`, the statement would be `SELECT COL26/1000 FROM JDETAB`.

8.4.1.2 Translating dates

J.D. Edwards stores dates as numeric values. The J.D. Edwards date format is similar to Julian date format except the year starts with 1900.

SAP Data Services handles J.D. Edwards numeric values by reading the date information and translating with an internal formula based on the J.D. Edwards start year. For example, the software would handle the date February 18, 1999 as follows:

Year = (1999 - 1900) * 1000;

Day = 31 + 18; [31 is the # of days in January]

The resulting Day value is the number of days since 1/1/1999. The Julian date value is Year + Day = `99049`.

Like decimal translation, there are three cases when the software does not automatically translate dates:

- SQL transform
- sql function
- pushdown_sql function

In these cases, you must use the `JDE_Date` function.

Related Information

JDE_Date [page 2360]

8.4.1.3 Translating time

SAP Data Services does not automatically interpret J.D. Edwards time data types. If you need to translate a number to its time value, you can use the `JDE_Time` function.

Related Information

JDE_Time [page 2361]

8.4.2 Functions

When conversion is not automatic, you can use SAP Data Services functions to convert J.D. Edwards data to internal SAP Data Services data types:

- JDE_Date
- JDE_Time

You can access these functions from the query editor: go to the *Mapping* or the *Where* tabs, and click the *Functions* button.

> i Note
>
> You cannot use the lookup function with a J. D. Edwards datastore. Use the lookup_ext function instead.

8.4.2.1 JDE_Date

Given a Julian date stored in a J.D. Edwards database, `JDE_Date` returns the equivalent value as a SAP Data Services date data type.

Syntax

```
JDE_Date(<jde_julian_date>)
```

Return Value

date: The SAP Data Services date data type equivalent to the specified Julian date.

Where

`<jde_julian_date>` : The integer column in the input table that contains a J.D. Edwards Julian date.

Example

Function	Results
`JDE_Date (99049)`	`2/18/1999`

You can translate J.D. Edwards Julian dates using column mapping. When you input a J.D. Edwards Julian date number (for example, `99049`), an SAP Data Services date type results (for example, `2/18/1999`). Enter the function in the *Mapping* section of the Query transform editor.

Use this function when the software does not automatically translate dates.

Note

Because this function takes an integer as input, map the incoming column to an integer column before applying the function.

Related Information

Translating dates [page 2359]

8.4.2.2 JDE_Time

Given a number representing time in J.D. Edwards, `JDE_Time` returns an SAP Data Services character value that represents the equivalent military time in **`<HH24:MI:SS>`** format (where **`<HH>`** is hours, **`<MI>`** is minutes, and **`<SS>`** is seconds).

Syntax

```
JDE_Time(<time_integer>)
```

Return Value

Char(8): The time in **`<HH>`** `24`: **`<MI>`** : **`<SS>`** format.

Where

`<time_integer>` : An integer column in the input table that represents a J.D. Edwards time.

Example

Function	Results
`JDE_Time(92513)`	`09:25:13`

You can translate times from J.D. Edwards formatting into SAP Data Services formatting using column mapping. For instance, if you input a J.D. Edwards record update time of 92513, the resulting SAP Data Services time would be 9:25:13. Enter the function in the *Mapping* section of the Query transform editor.

9 Supplement for Oracle Applications

9.1 Introduction

The SAP Data Services Oracle Applications interface allows you to create Oracle Application datastores and import tables for use as sources in SAP Data Services jobs. With this interface, you can use the Designer to:

- View objects such as tables, functions, and procedures in a database instance under a specified Oracle Applications environment.
- Import metadata for Oracle Applications objects into SAP Data Services.
- Create data flows that use Oracle Applications objects (such as tables) as sources and targets.

9.2 System requirements

The Oracle Applications interface allows you to connect SAP Data Services with Oracle Applications for importing metadata.

These applications operate on an underlying Oracle database. Install the drivers you need to connect Oracle with SAP Data Services on the same computers on which you install the Designer and Job Server components.

The Oracle Applications interface supports version 11.5 and later versions.

The Oracle Applications interface can be installed using the SAP Data Services installer.

Because Data Services connects to Oracle through a database connection, the software can see, import, and use all Oracle Applications objects that the provided Oracle database user (such as APPS) can access without requiring application user login security information.

The Oracle Applications interface uses Oracle Application object library tables (which use the prefix `FND_%`) to get metadata information. For more information about Oracle Application metadata, see your Oracle Application Object Library/Workflow Technical Reference Manual.

9.3 Datastores

SAP Data Services uses datastore connections to link with other applications or databases. In a design environment, you use datastores to browse, search, or import metadata that represents external tables, files, messages, and other database objects. When running jobs, the software uses datastore information to move data between source and target databases and applications.

After you define an Oracle Applications datastore in SAP Data Services, you can browse application modules, tables, table descriptions, and column descriptions. While stored procedures and functions are also displayed, they are not grouped by application because they are not application-specific.

You must define Oracle Applications datastore connections with accurate Oracle Applications information to ensure the accessibility of the tables.

9.3.1 Defining an Oracle Applications datastore

With an Oracle Applications datastore connection, you can browse Oracle Applications metadata.

9.3.1.1 To define an Oracle Applications datastore

1. In the object library of the Designer, go to the *Datastores* tab.
2. Right-click inside the object library window and select *New*.
3. In the *Datastore Editor* window, enter a name for this datastore (`DS_OracleApps`, for example).
4. In the *Application type* list, select *Oracle_Applications*.
5. Enter the connection information.
6. Click *OK*.
   The Oracle Applications datastore appears in the object library.

9.3.1.2 Oracle datastore options

The following table describes the options specific to Oracle Applications.

Oracle Applications option	Possible values	Description
Process flexfield names	Yes, No	Determines whether SAP Data Services should use descriptive names for flexfield columns based on the flexfield information. If set to No, the software uses database column names for all columns including those that belong to flexfields.
Schema owner	Refer to the requirements for your application	Type the schema owner name for foundation tables (the name is usually **`apps`**).

9.3.2 Browsing and importing metadata

After creating an Oracle Applications datastore, you can browse and import the metadata from the connected application.

i Note

Although you cannot browse packages (there are too many), you can import them by name.

9.3.2.1 To browse and import the metadata

1. View the modules in the Oracle Application. You can:
    - Right-click the datastore name and select *Open*, or
    - Double-click the datastore name

   The workspace displays the list of application modules.
2. View the tables in a particular module. You can:
    - Expand the module tree, or
    - Double-click a module (folder)
3. To import table data, right-click a table name and select *Import*.

   > Note
   >
   > When you import a table into SAP Data Services via the Oracle Applications interface, the software does not preserve the table hierarchy. All tables are listed at the same level.

You can also import tables using the Designer's *Import by Name* feature.

9.3.3 Extracting data from Oracle Applications systems

When extracting data from Oracle Applications, SAP Data Services processes and converts data types appropriately.

9.4 Flexfields

SAP Data Services processes both descriptive flexfields and key flexfields from Oracle Applications if they are in use. Furthermore, Data Services can distinguish between context-specific and non-context-specific flexfields. The meaning of a context-specific flexfield is based on the value of another (context or structure) field.

Columns created from flexfields have column names based on the physical database name or end-user column name for that flexfield.

For consistency, when the software bases a column name on an end-user column name, it will translate the end-user column name from lowercase letters to uppercase, replacing all non-alphanumeric characters with an underscore.

If the physical name of a flexfield is descriptive (like `TAX_CODE_FLAG`), the software displays that physical name as the column name in a schema. However, if the physical name is generic (for example, `ATTRIBUTE1` or `SEGMENT1`) and associated with a descriptive end-user column name, the software truncates the end-user column name to 50 characters, appends it with an alphanumeric suffix, and uses it as the column name.

The software appends column names created from descriptive flexfield end-user column names with a `_D#` while column names created from key flexfields are appended with a `_K#` where `#` is a unique number. For example, if the physical name of a descriptive flexfield is `ATTRIBUTE1` and the end-user column name is `Status code`, the logical name might display as `STATUS_CODE_D6`.

When a descriptive flexfield's end-user column name is context-specific, representing multiple names, the first end-user column name (with suffix identifier) appears as the column name and a note appears in the key flexfield's column description, structured as follows:

```
<first end-user
             column name> (multi-flex column - multiple structures based on
          <content field COLUMN
             NAME> =
          <"content value>"
             - <field comment> )
```

When a key flexfield's end-user column name is structure-specific, representing multiple names, the first end-user column name (with suffix identifier) appears as the column name and a note appears in the key flexfield's column description, structured as follows:

`<first end-user column name>` (multi-flex column - multiple structures based on `ID_FLEX_CODE=` `"`**`<structure code value>`**`"ID_FLEX_NUM-` **`<column name>`** `=` **`<structure number>`** `-` **`<field comment>`**)

The following table shows how flexfield column names for both descriptive and key flexfields are based on physical name and end user column name:

Table 184: Descriptive Flexfields

From Oracle Applications		To SAP Data Services	
Physical name:	**End-user column name:**	**Column name:**	**Business name:**
STAT_CODE	N/A	STAT_CODE	STAT_CODE
ATTRIBUTE1	Status code	STATUS_CODE_D1	Status code
ATTRIBUTE1	-(None)	ATTRIBUTE1	-(None)

Table 185: Key Flexfields

From Oracle Applications		To SAP Data Services	
Physical name:	**End-user column name:**	**Column name:**	**Business name:**
STAT_CODE	N/A	STAT_CODE	STAT_CODE
SEGMENT1	Status code	STATUS_CODE_K1	Status code
SEGMENT1	-(None)	SEGMENT1	-(None)

i Note

Using flexfields with SQL function calls is not supported and causes an invalid column name error. Instead, use physical column names in functions that pass commands directly to the database such as SQL and PUSHDOWN_SQL.

To learn more about flexfields, see the *Oracle Applications Flexfields Guide*.

10 Supplement for PeopleSoft

10.1 Introduction

Welcome to the *SAP Data Services Supplement for PeopleSoft*. This guide contains information about how to use Data Services with PeopleSoft.

10.1.1 Using Data Services in a PeopleSoft environment

If you install the PeopleSoft interface, you can use PeopleSoft HRMS and ERP application data as a data source in SAP Data Services.

Through the PeopleSoft datastore you define, you can navigate through PeopleSoft metadata external to SAP Data Services (using standard PeopleSoft panels and menus) and import metadata for PeopleSoft source tables into the repository. The local object library then displays the datastore objects including PeopleSoft domains, hierarchies, and tables.

In addition, capabilities in the software that support PeopleSoft include the following:

- You can view valid domain values for a column from within the query transform.
  This allows you to filter on specific domain values for a column.
- The software can determine whether a given value for a column is valid within the domain of the column.
  The software flags those that are not valid.
- The software allows extraction of data based on effective dates without requiring you to write a secondary query. This functionality is particularly useful in implementing data warehouses containing HR data—it significantly reduces the complexity and increases the performance of extracting effective-dated data.
- The software allows extraction of the hierarchical data in PeopleSoft trees.

10.2 PeopleSoft Datastores

With the PeopleSoft interface, you can use a PeopleSoft system as an SAP Data Services source. To use a PeopleSoft data source, you must:

- Define a datastore that will serve as the logical link to your PeopleSoft system
- Import your PeopleSoft metadata into the datastore

Related Information

Designer Guide: Datastores [page 209]

10.2.1 Defining PeopleSoft datastore

You must define a PeopleSoft datastore in SAP Data Services when you are extracting data from or loading data to PeopleSoft.

1. Go to the *Datastores* tab of the object library.
2. Right-click inside the object library window and choose *New*.

   The *Create New Datastore* window appears.
3. Enter a name for the new datastore in the *Datastore name* box.

   You can give the datastore any name you want, and you can change the name later if necessary. The name can contain any alpha or numeric characters or underscores (_). It cannot contain spaces.
4. In the *Datastore type* box, choose *PeopleSoft*.
5. In the *Database type* box, choose the appropriate database.
6. Enter the appropriate information for the database type you selected.
7. Click *OK*.

   The software creates the datastore and it appears in the object library window.

Related Information

Reference Guide: Datastore [page 860]

10.2.2 Browsing PeopleSoft metadata

After you create the datastore, you can access the PeopleSoft metadata via the SAP Data Services datastore explorer, which displays information in a tree format.

The information displayed consists of metadata related directly to database data—no information about calculation fields, images, buttons, or other GUI-related items is included.

Browsing PeopleSoft data in the software is a lot like browsing in PeopleTools—icons in the display represent PeopleSoft menu groups, menus, menu bars, menu items, panels, and panel fields. Each panel field displays the caption, table column, and table name.

Icon	Metadata type	Description
🗔	Menu Group	The entire set of commands available in PeopleSoft applications for a specific database.
🗔	Menu	A set of commands for a specific application.
🗏	Menu Bar	The top level of the menu.

Icon	Metadata type	Description
▤	Menu Item	The commands that make up each menu bar.
▤	Panel	A vehicle for capturing and displaying data. Each panel can have multiple sub-panels, each represented by the same icon.
▭	Panel Fields	Represents a column on a table or in a specific view.

10.2.2.1 To browse PeopleSoft metadata

1. In the object library, go to the *Datastores* tab.
2. Right-click the PeopleSoft datastore name and select *Open*.

   The datastore explorer window opens.

10.2.3 Importing PeopleSoft metadata

To access PeopleSoft metadata from SAP Data Services, you must import the metadata into the object library.

You can import tables, PeopleSoft trees (called *hierarchies* in SAP Data Services), and domains.

You can import PeopleSoft metadata in one of three ways:

- By name
- By browsing
- By searching

10.2.3.1 To import PeopleSoft metadata by name

1. In the object library, go to the *Datastore* tab.
2. Right-click the datastore name and select *Import By Name*.
3. In the Import by Name dialog box, specify the *Type* of the item to import.

   To import a PeopleSoft tree, select Hierarchy.
4. In the *Name* box, enter the name of the item to import.

   The default is to import domain information automatically with any tables you import. You can choose not to import the associated domains by deselecting the *Import associated domains* option.
5. To import a hierarchy, enter the tree name in the *Name* box and enter the Category and Structure associated with the hierarchy.
6. Click *OK*.

   The information appears in the object library.

10.2.3.2 To import PeopleSoft metadata by searching

You can search for PeopleSoft items as you would any items to be imported into the software. That is, right-click the datastore name in the object library and choose *Search*.

Related Information

Designer Guide: Importing metadata through a database datastore [page 219]

10.2.3.3 To import PeopleSoft metadata by browsing

1. In the object library, go to the *Datastores* tab.
2. Right-click the datastore name and choose *Open*.

   The datastore explorer opens in the workspace and lists the available tables, domains, and hierarchies.
3. Right-click the names of the items you want to import and choose *Import*.

   To import a table, select a name at the panel field level.

   To import a hierarchy, select a name at the tree level.

Related Information

Browsing PeopleSoft metadata [page 2369]

10.2.4 Metadata for PeopleSoft domains

This section discusses metadata for PeopleSoft domains.

Related Information

What is a domain? [page 2372]
Using PeopleSoft domains in SAP Data Services [page 2372]

10.2.4.1 What is a domain?

A domain is a lookup table that pairs a coded value with a textual description of the value. Domain values are typically used to specify column data where the possible range of values is constrained within a particular set.

Using a domain value instead of the text description can save space when the number of records is large. However, when working with the table to create queries, you might prefer to see a text description as opposed to looking for the appropriate code for the source data.

10.2.4.2 Using PeopleSoft domains in SAP Data Services

Domain values in PeopleSoft are available to end users only through PeopleSoft panels.

SAP Data Services has built-in capabilities that make it much easier to deal with domain values in data movement applications.

You browse, search, and import domain metadata in the same way you browse, search, and import any PeopleSoft metadata.

PeopleSoft tables have columns that should be constrained to contain only values from a particular set (that is, a set within a domain). This set of possible domain values is in another table in the PeopleSoft database called XLATTABLE.

When importing table metadata from a PeopleSoft database, the software can automatically import associated domain data to make it easier to work with the values that appear in rows from the imported table. By default, the software imports associated domain data, but you can prevent this by deselecting the *Import associated domains* check box on the Import By Name window.

When importing domain values, the repository only stores the currently effective value. The currently effective value is determined using the date when the import takes place.

Imported domains appear nested under the datastore in the object library.

Related Information

Browsing PeopleSoft metadata [page 2369]
Importing PeopleSoft metadata [page 2370]
Using PeopleSoft domains in data flows [page 2375]

10.2.5 Metadata for PeopleSoft trees

You can perform the same kind of business model navigation and data browsing on PeopleSoft trees (called hierarchies in Data Services) that you can on other objects.

You browse, search, and import hierarchy metadata similarly to the way you browse, search, and import any PeopleSoft metadata.

Imported hierarchies appear nested under the datastore in the object library.

Related Information

Browsing PeopleSoft metadata [page 2369]
Importing PeopleSoft metadata [page 2370]
Extracting PeopleSoft tree data [page 2377]

10.3 Data Flows with PeopleSoft Data

Data flows extract, transform, and load data. This section describes information unique to data flows that extract, transform, and load PeopleSoft data.

Related Information

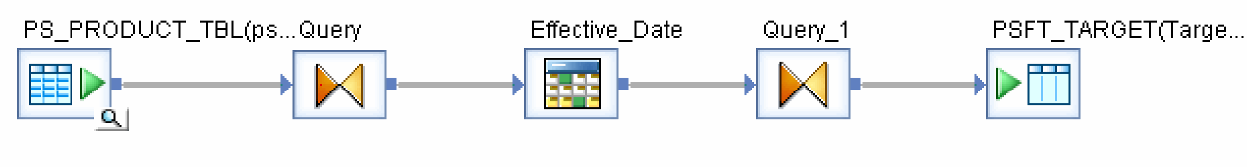
Designer Guide: Data Flows [page 281]

10.3.1 Using effective dates from PeopleSoft data

This section provides an example that shows how you can use effective dates when populating a dimension table in the product group. This example includes several operations:

- Extracts data from table `PS_PRODUCT_TBL`
- Selects a subset of columns for the target
- Filters the data based on status
- Creates effective date ranges using the Effective Date transform
- Retrieves only the rows effective on a particular date
- Loads the data into the target

One data flow completes these operations.



10.3.1.1 Selecting a subset of source columns

The first query selects a subset of the columns:

```
PS_PRODUCT_TBL.EFF_STATUS = 'A'
```

10.3.1.2 Filtering based on status

The first query includes a `WHERE` clause that limits the products selected to those with a status of `Active`. The status information comes from the domain values in the `EFF_STATUS` column in the input.

Related Information

Using PeopleSoft domains in data flows [page 2375]

10.3.1.2.1 To display the relevant domain values for the EFF_STATUS column

1. Click *Domains* in the `WHERE` tab of the query editor. The Picker window opens and shows a list of domains and descriptions.
2. Select the `EFF_STATUS` domain and click the *Show Values* button (second button at the top of the Picker window) to see the available domain values. The domain values appear to the right of the descriptions.
3. To put the domain value in the `WHERE` clause,
   a) Drag the `EFF_STATUS` column from the source schema into the `WHERE` clause.
   b) Enter an equal sign (=) after the column name.
   c) Place the cursor where the value needs to appear in the *WHERE* tab and click the description name *Active* in the Picker window.

10.3.1.3 Creating effective date ranges

This example retrieves only those product groups that are valid today. There are three steps involved:

1. Retrieve the effective-from date (`EFFDT` column) from the source.
2. Generate effective-to dates for the source rows using the Effective Date transform.
3. Filter out rows that are not effective on today's date.

Related Information

Reference Guide: Transforms [page 1067]

10.3.1.4 Retrieving records with current effective dates

The second query in the data flow contains a `WHERE` clause that selects:

- Those rows where the effective-from date from the source (`EFFDT`) is less than or equal to the system date AND
- Those rows where the effective-to date from the results of the Effective Date transform is greater than the system date

The query editor contains the `WHERE` clause.

```
Effective_Date.EFFDT <= sysdate( ) and
Effective_Date.EFFECTIVE_TO_COLUMN > sysdate ( )
```

10.3.1.5 Using PeopleSoft domains in data flows

SAP Data Services helps you:

- Build queries containing domain values
- Substitute domain descriptions for cryptic domain values
- Build queries where you need to know some domain values, for example while filtering
- Validate imported data that contains domain values

The properties for a given column enable you to explicitly associate a given column with a given domain.

You can assign any available domain to any column regardless of data type.

Related Information

Metadata for PeopleSoft domains [page 2371]

10.3.1.5.1 To assign a domain to a column

1. Right-click a table name in the object library and select *Open*.
2. To open the column properties, right-click a column name in the table metadata and select *Properties*.
3. The drop-down list in the *Associated domain* text box displays the available domains.

Any available domain can be assigned to any column regardless of data type.

10.3.1.6 Using the Picker window

To open the domain Picker window, click the *Domains* button in a query editor that has a table as a source.

The four buttons at the top of the Picker window control the content of the window and initiate actions.

	Click to toggle between displaying "all" and "relevant" domains. All domains (default) are listed in alphabetical order; relevant domains are shown in the order in which they are used in the table or tables you have selected.
	Click to display the domain values in the window. Click again to hide the domain values.
#	Click to toggle between pasting only the domain value (default) and both the value and a commented description in the `WHERE` clause when you select the value.
	Pastes all selected values in the `WHERE` clause. Descriptions are not included. To enable this button, use Ctrl-clicks to select more than one domain description.

10.3.1.6.1 To include a column and a related domain value in a WHERE clause

1. Drag the column from the source schema into the `WHERE` clause.
2. Enter an equal sign (=) after the column name.
3. Enter a value or paste the value from the Picker window.

   Click Domains to open the Picker window.

10.3.1.7 Validating domains

Capabilities built into SAP Data Services allow you to validate data that contains domain values:

- value IN domain clause (part of a `WHERE` clause)
  This clause enables you to ensure that all rows in a table have a valid value for a domain column.
  Syntax:
  value IN datastore_name.owner.domain_name
  This clause returns TRUE if the value is a member of the domain specified.
  If the clause is part of a job being executed, the software searches the domain values in the XLATTABLE.
- `get_domain_description` function
  This function returns the description for a domain name. The description is returned as a quoted string.
  Syntax:

```
get_domain_description
(<'datastore_name..domain_name',>
< table_name.domain_name>)
```

10.3.1.8 Restrictions using domains

Some restrictions involving domains include:

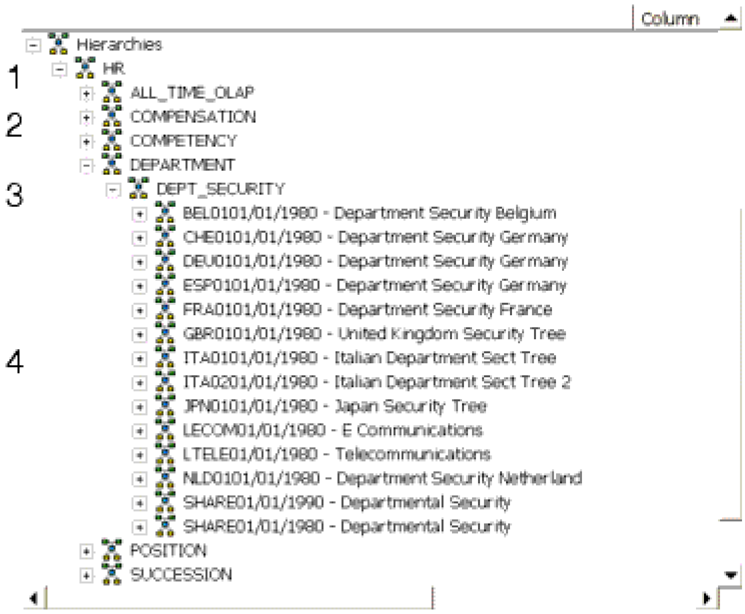
Prompt tables are not supported. (A prompt table is a user-defined table similar to the XLATTABLE in a database.)

Prior to validation at execution time, there is no type checking when a domain is associated with a column.

10.3.2 Extracting PeopleSoft tree data

In this example, SAP Data Services extracts data from a hierarchy (tree) and loads it into a table.

You can browse hierarchy information when you open a datastore. Expanding the hierarchy node in the workspace shows the tree levels. Consider the following example of an expanded hierarchy node in the workspace with *External Metadata* selected.

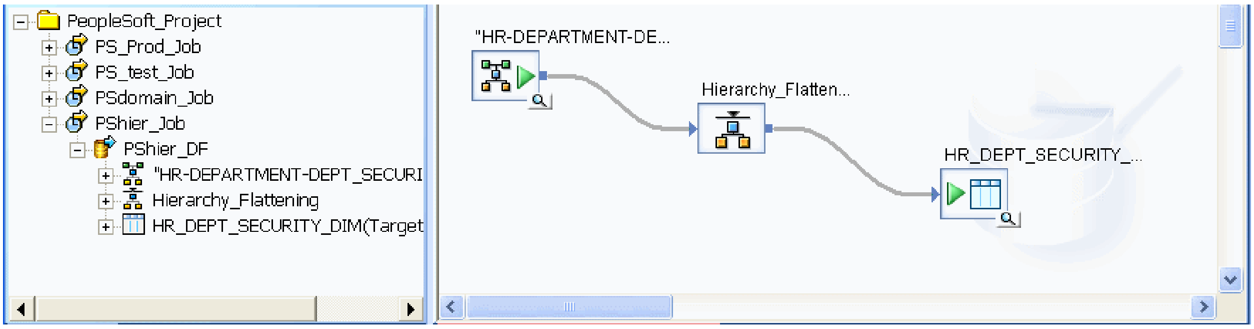


```
Hierarchies
  HR
    ALL_TIME_OLAP
    COMPENSATION
    COMPETENCY
    DEPARTMENT
      DEPT_SECURITY
        BEL0101/01/1980 - Department Security Belgium
        CHE0101/01/1980 - Department Security Germany
        DEU0101/01/1980 - Department Security Germany
        ESP0101/01/1980 - Department Security Germany
        FRA0101/01/1980 - Department Security France
        GBR0101/01/1980 - United Kingdom Security Tree
        ITA0101/01/1980 - Italian Department Sect Tree
        ITA0201/01/1980 - Italian Department Sect Tree 2
        JPN0101/01/1980 - Japan Security Tree
        LECOM01/01/1980 - E Communications
        LTELE01/01/1980 - Telecommunications
        NLD0101/01/1980 - Department Security Netherland
        SHARE01/01/1990 - Departmental Security
        SHARE01/01/1980 - Departmental Security
    POSITION
    SUCCESSION
```

1. Category
2. Structure
3. Tree
4. Set

You import hierarchical data at the tree level. To import the tree into SAP Data Services, select the tree DEPT_SECURITY, right-click, and choose *Import*. After you import the tree, it appears as a hierarchy in the object library.

The sample job PShier_Job consists of a single data flow that extracts the data from the source hierarchy object HR-DEPARTMENT-DEPT_SECURITY and loads it into a table.



Double-clicking the source hierarchy object opens the editor to show the options available for it.

In the hierarchy editor, you must specify:

- All set IDs, a specific set ID, or multiple set IDs.
  Select the *All set IDs* check box to obtain all the set IDs associated with this hierarchy. Each set ID identifies a subset of a tree that groups data that have similar values or rules. In this example, the tree contains a set of data for Belgium (BEL01), a set for France (FRA01), and so forth.
  Clear the *All set IDs* check box to open the Set ID box where you can:
  - Enter the name of the specific set ID for which you want to extract data.
  - Enter multiple set IDs, separated by commas. For example:

    ```
    BEL01, FRA01
    ```
- Date options
  - Current date
    Select Current date to extract data that is effective as of the date returned by the sysdate function.
  - All dates
    Select All dates to extract data with all effective dates for the set IDs that you specified.
  - Snapshot date
    Select Snapshot date to extract data that was effective on a specific date. Specify a date in the past or future in one of the following ways:
    - Date
      Select Date in the drop-down list to specify the date as a four-digit year, a period, a two digit month, a period, and a two-digit day (YYYY.MM.DD). You can select each part of the date and use the arrows to increment or decrement each value.
    - Variable
      Select Variable in the drop-down list to specify a varchar variable name that contains a valid date in the yyyy.mm.dd format.
      You can select one of the variable names in the drop-down list. Alternatively, you can type in the name of a new variable, but you must define it in the Variables and Parameters window (▶ *Tools* ▶ *Variables* ▶).
      If you do not enter a variable name, the Snapshot date defaults to the current date.

> Note
>
> You can use a local or global variable for the date. Because the variable can be a global variable, validating the data flow or work flow does not check for invalid data types for this variable. Be sure to validate the Job to check for invalid data types for this date variable.

Before you load the tree data into the table, you can flatten it with the Hierarchy_Flattening transform. On the transform editor:

- Choose to flatten the tree horizontally or vertically.
- Specify the *Parent column* and *Child column* to identify the columns in the source data that contain the parent identifier and child identifier in each parent-child relationship.
- Select *View* > *Refresh* or press F5 to refresh the target schema after you identify the source columns.

Related Information

PeopleSoft Datastores [page 2368]
Hierarchy [page 2382]

10.4 Reference Information

This section contains reference information specific to the PeopleSoft interface for SAP Data Services.

Specifically, this section describes SAP Data Services objects specific to PeopleSoft and SAP Data Services objects with supplemental information for the PeopleSoft interface.

This section contains information about the following objects:

Object	Class	Description
Datastore	Single-use	Specifies the connection information Data Services needs to access a database or other data source. Cannot be dropped.
Domain	Reusable	A lookup table that pairs a coded value with a textual description of the value. Domain values are typically used to specify column data where the possible range of values is constrained within a particular set.
Hierarchy	Reusable	Browse, search, and import hierarchy metadata.
Query	Single-use	Retrieves a data set that satisfies conditions that you specify.

10.4.1 Datastore

Class

Reusable

Access

In the object library, click the *Datastores* tab.

Description

A datastore provides a connection to a data source such as a database. Through the datastore connection, SAP Data Services is able to import descriptions of the data source such as its metadata. When you specify tables as sources or targets in a data flow, the software uses the datastore to determine how to read data from or load data to those tables. In addition, some transforms and functions require a datastore name to qualify the tables they access.

If you delete a datastore from the object library, you must remove references to the datastore from the following locations:

- Source or target tables using this datastore in your diagrams
- The `lookup` and `key_generation` functions and Key_Generation, History_Preserving, Table_Comparison, and SQL transform references

Datastores have the following properties:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Text that you enter to describe and document the datastore.

After you create a datastore, you can import metadata about the objects, such as tables and functions, in that datastore.

PeopleSoft datastores

Set the following options to define a PeopleSoft datastore:

Option	Description
Name	Specify the datastore name. The software uses this name to reference the datastore from other object definitions.
Datastore type	Choose PeopleSoft to display the options for PeopleSoft datastores. You cannot edit this option after creating the datastore.
Database type	Select either *Microsoft_SQL_Server* or *Oracle* to indicate the database type used as the database layer of your PeopleSoft application server. The remaining options in the datastore definition are specific to the database type.

Related Information

Designer Guide: Datastores [page 219]
Reference Guide: Database datastores [page 868]

10.4.2 Domain

Class

Reusable

Access

- To view the list of imported domains and domain values from the object library, click the *Datastores* tab.
- Use domain values from inside query transforms with inputs from tables associated with domains. Click *Domains* on the *Where* tab to open a window that helps you pick domain values.

Description

A domain is a lookup table that pairs a coded value with a text description of the value. You will typically use domain values to specify column data where the possible range of values is constrained to a particular set.

Domains have two built-in attributes.

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	The description of the domain imported from the application or database.

View domain values, descriptions of the values, and the effective dates for each value by opening the domain from the object library.

10.4.3 Hierarchy

Class

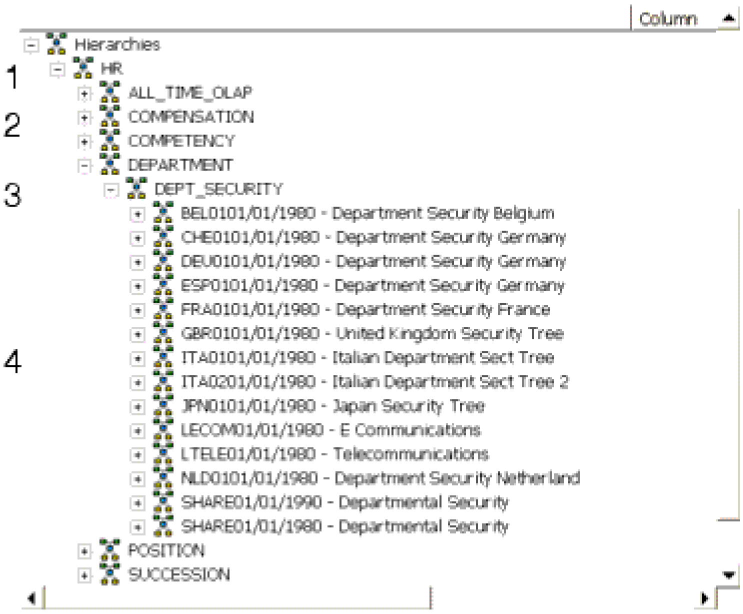
Reusable

Access

In the object library, click the *Datastores* tab and expand a datastore listing (click the plus sign next to the datastore name).

Description

A hierarchy is a PeopleSoft tree.



```
Hierarchies
  HR
    ALL_TIME_OLAP
    COMPENSATION
    COMPETENCY
    DEPARTMENT
      DEPT_SECURITY
        BEL0101/01/1980 - Department Security Belgium
        CHE0101/01/1980 - Department Security Germany
        DEU0101/01/1980 - Department Security Germany
        ESP0101/01/1980 - Department Security Germany
        FRA0101/01/1980 - Department Security France
        GBR0101/01/1980 - United Kingdom Security Tree
        ITA0101/01/1980 - Italian Department Sect Tree
        ITA0201/01/1980 - Italian Department Sect Tree 2
        JPN0101/01/1980 - Japan Security Tree
        LECOM01/01/1980 - E Communications
        LTELE01/01/1980 - Telecommunications
        NLD0101/01/1980 - Department Security Netherland
        SHARE01/01/1990 - Departmental Security
        SHARE01/01/1980 - Departmental Security
    POSITION
    SUCCESSION
```

A hierarchy contains the following levels:

1. Category: A category represents a major business organization. The example datastore explorer displays the category *HR*.
2. Tree structure: A tree structure represents a department or group within a category. The example datastore explorer displays three tree structures: *COMPENSATION*, *COMPETENCY*, and *DEPARTMENT*.
3. Tree: A tree represents a specific type of data within a tree structure. The example datastore explorer shows the *DEPT_SECURITY* tre expanded. You import hierarchical data at the tree level.
4. Set: A set is a subset of the tree that contains data that have similar values or rules. Each set is identified by a set ID. The above datastore explorer shows set IDs *BEL01*, *DEU01*, and so forth, and each set contains data specific to an organization.

To extract hierarchical information, select a hierarchy type or tree, import its metadata into your repository, then drag the hierarchy icon from the object library into a data flow.

From the datastore explorer, you can choose the hierarchy to import.

You can also import a tree by specifying the tree by name. Select the datastore in the object library, right-click, and choose *Import By Name*.

Choose *Hierarchy* from the *Type* drop-down menu.

Imported tree

After you import the tree, it appears as a hierarchy object in the object library nested under the datastore name.

The hierarchy object has the following properties:

Property	Description
Name	The name of the hierarchy object. The name is constructed as follows: `category-structure-tree(datastore)` The name (except the datastore name) is case sensitive.
Description	The description of the tree as included in the PeopleSoft table.

The hierarchy object has the following attributes, the values of which are included in the metadata imported for the hierarchy.

Attribute	Description
Category	Tree_Node_Description
Structure	Tree_Node_Record_Primary_Key1 -Tree_Node_Record_Primary_Key9
Set_ID	
Snapshot_Date	
Tree_Name	Tree_Leaf_Field_Name
Tree_with_Leaf?	Tree_Leaf_Description
Tree_Node_Record_Name	Tree_Leaf_Record_Primary_Key1 - Tree_Leaf_Record_Primary_Key9
Tree_Node_Field_Name	

Selecting the hierarchy in the object library and choosing *Open* displays the object properties and the source schema that results from the hierarchy extraction.

Source column name	Description
NodeID	The parent value in the relationship described by this row.
Description	The parent description.
ChildID	The child value in the relationship described by this row.
ChildDesc	The child description.
MaxDepth	The maximum number of nodes between the root node and the lowest node in the hierarchy.
TreeLevel	The level in the hierarchy that this row describes.
Category	The category in which this hierarchy is included.
Structure	The tree structure in which this hierarchy is included.
SETID	The identification of the subset of the tree. Each set groups together data that have similar values or rules.
SnapshotDate	The date when the hierarchy data was extracted.
EFFDT	The date when the data in a set became or will become effective.

Hierarchy instance

You can drag a hierarchy from the object library into a data flow definition.

The hierarchy editor displays the datastore information for the hierarchy and provides options for the instance of the hierarchy object:

Option	Description
All set IDs or *Set ID*	*Set ID* selects one or more trees out of the structure imported into SAP Data Services. Use commas to separate multiple set IDs. The set ID identifies a subset of the tree that groups together data that have similar values or rules. It appears as part of the name of the level below the level of hierarchy structure imported into the software. *All set IDs* selects all versions of a tree.
Current date or *All dates* or *Snapshot date*	*Current date* selects only the data that has an effective date that is equal to the date returned by the sysdate function. *All dates* selects data with all effective dates. *Snapshot date* filters the extracted values by the effective date you specify in one of the following ways: ● *Date* allows you to specify the date as a four-digit year, a period, a two digit month, a period, and a two-digit day (YYYY.MM.DD). You can select each part of the date and use the arrows to increment or decrement the digits. ● *Variable* allows you to specify a varchar variable name that contains a valid date in the yyyy.mm.dd format You can select one of the variable names in the drop-down list. Alternatively, can type in the name of a new variable, but you must define it in the Variables and Parameters window (▶ *Tools* ▶ *Variables* ◀). If you do not enter a variable name, the Snapshot date defaults to the current date. ℹ Note You can use a local or global variable for the date. Because the variable can be a global variable, validating the data flow or work flow does not check for invalid data types for this variable. Be sure to validate the Job to check for invalid data types for this date variable.

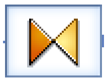
Related Information

Importing PeopleSoft metadata [page 2370]

Extracting PeopleSoft tree data [page 2377]

Importing PeopleSoft metadata [page 2370]

10.4.4 Query



Class

Single-use

Access

With a data flow diagram in the workspace, click the query icon in the tool palette, then click in the workspace.

Description

A Query transform, like a SQL SELECT statement, retrieves a data set that satisfies the conditions you specify. With a Query transform, you can:

- Map columns from input to output schema
- Add new columns, nested schemas, and functions to the output schema
- Choose the data to extract
- Perform operations on the data
- Join data from multiple sources

Editor

From tab: Use the *From* tab to specify joins and set join conditions.

Where tab: Use the *Where* tab to restrict the result set.

With the PeopleSoft interface, you can use the *Domains* button. Domains constrain retrieved data sets. Click the *Domains* button to open the Picker window, which helps you build an expression using a domain.

Related Information

Reference Guide: Transforms, Query transform [page 1441]

Metadata for PeopleSoft domains [page 2371]

11 Supplement for Salesforce.com

11.1 Introduction

This user guide tells you how to use the SAP Data Services Adapter for Salesforce.com interface to integrate Salesforce.com with Data Services.

The Data Services Adapter for Salesforce.com interface allows you to create a datastore that connects to the Salesforce.com web service and retrieves data using Data Services data flows.

11.1.1 Audience and assumptions

This user guide assumes the following:

- You understand how to use the most current version of SAP Data Services to design and run batch and real-time data flows and administer Data Services processes. (Administer adapters from the Administrator.)
- You have a working knowledge of Salesforce.com..
- You know what an adapter is and the role it plays in business systems integration.
- You are familiar with how to use SQL query statements.
- You understand Changed Data Capture concepts.
- You are familiar with object-oriented modeling and can work with an object-oriented XML configuration file.
- Because you will integrate Data Services, the Data Services Adapter for Salesforce.com, and Salesforce.com, familiarity with systems administration and systems integration issues is recommended.

11.2 Overview and installation

11.2.1 Installing the Adapter for Salesforce.com

The SAP Data Services Adapter for Salesforce.com is automatically installed when you install Data Services version 12.0.0 or later. The adapter is associated with several files including:

- Adapter jar files
- Adapter configuration templates
- Salesforce.com Software System extensions
- *Supplement for Salesforce.com* (this document)

11.2.1.1 Requirements

The Job Server you associate with adapters must be configured to manage adapters. For general Job Server installation and configuration information, see the *SAP Data Services Installation Guide*.

Note

For information about Salesforce.com, visit the Salesforce.com Web page.

Related Information

Deployment overview [page 2390]

11.2.2 Adapter overview

The SAP Data Services Adapter for Salesforce.com allows you to access Salesforce.com data from within the native Data Services extraction, transformation and loading (ETL) environment. The adapter interface allows you to quickly and easily take advantage of Salesforce.com by:

- Supporting a fully automated process for Salesforce.com configuration
- Allowing you to browse Salesforce.com schema metadata in the same manner as all sources and targets from within the Designer interface

11.2.2.1 To use the Adapter for Salesforce.com from SAP Data Services

1. Install SAP Data Services version 12.0.0 or later (for more information, see the *Installation Guides*). Installing the software automatically installs the Salesforce.com adapter.
2. Configure the Job Server local to your installation of the software for adapter management. See the *Installation Guides* and the *Administrator Guide* for Job Server configuration details.
3. Configure the Data Services Adapter for Salesforce.com interface.

   Configure one or more adapter instances. You can configure and use multiple instances simultaneously.
4. Through the Designer, use the adapter inside data flows. You can:
   - Create the adapter's datastore
   - Import the adapter's metadata
   - Use imported metadata as sources in your data flows
   - Run jobs and verify results

11.3 Deploying the Adapter

This section explains the actions required to deploy the SAP Data Services Adapter for Salesforce.com interface. Tasks are sequenced in logical order of performance. However, you may need to modify the sequence based on your environment.

11.3.1 Deployment overview

All Data Services adapters communicate with Data Services through a designated Adapter Manager Job Server. An adapter must be installed on the same computer as this Job Server before you can integrate the adapter with the software using the Administrator and Designer. After the adapter is installed:

1. Use the Server Manager utility to configure adapter connections with the Adapter Manager Job Server. For details, see the "Server management" section in the *Data Services Administrator Guide* as well as the "Adapters" section in the *Data Services Management Console Guide*.
2. From the Administrator:
   - Configure an adapter instance.
   - Start and stop the adapter instance.
3. From the Designer:
   - Create the datastore in the object library. The datastore and adapter make it possible for you to import metadata from Salesforce.com into the software.
   - Browse and import metadata through the datastore. Use metadata accessed through the adapter to create batch and/or real-time jobs. For details, see the "Adapter datastores" section of the *Data Services Designer Guide*.
   - Design flows that move Salesforce.com data through the applications you design using the software.
   - Run applications to finalize the integration process (includes troubleshooting and parameter adjustments).

11.3.2 Configure the adapter

Integrate SAP Data Services with Salesforce.com by combining an instance of the Adapter for Salesforce.com with a data flow created in the Designer. To use an adapter instance, you must first configure it as described in this section. You can configure one or more adapter instances.

11.3.2.1 Configure an adapter instance

Use the Administrator to add an Adapter for Salesforce.com to the SAP Data Services system and to edit existing adapter configurations. Until you add the adapter in the Administrator, you cannot run jobs using information from that adapter.

Note

Before you add an adapter in the Administrator, you must first establish Administrator connection to your adapter-enabled repository. For general information on connecting repositories to the Administrator, refer to the "Administrator Management" section of the *Management Console Guide*.

11.3.2.1.1 To add an adapter instance in the Administrator

1. Select a Job Server name under the *Adapter Instances* node in the navigation tree. Or, select *Adapter Instances* > *Job Server*.
2. Select the Configuration tab.
3. On the Adapter Instance Configuration page, click *Add* to see a list of adapters managed by that Job Server.
4. Select the Adapter for Salesforce.com from the list, then click *Apply*.
5. Complete the Adapter Instance start-up configuration form.

Option	Description
Adapter Instance Name	(Required) Enter a unique name to identify this instance of the adapter.
Access Server Host	Leave blank (or you can provide the correct Access Server Host information). Note If you enter incorrect information in this text box, configuration will fail with an error message.
Access Server Port	Leave blank (or you can provide the correct Access Server Port information). Note If you enter incorrect information in this text box, configuration will fail with an error message.
Character Set	Converts text characters to and from bytes.
Adapter Retry Count	Applies if the adapter instance fails or crashes. Enter 0 for no retries and a negative number for indefinite retries.
Adapter Retry Interval	Wait in milliseconds between adapter retry attempts.
Classpath	Indicates the -classpath Java parameter value when the adapter starts.
Autostart	When set to True, the adapter interface automatically starts when the Administrator starts.
Trace mode	Set this flag to control the number of trace messages the adapter writes. There are two settings: True Adapter interface writes additional information messages to help debug problems. False Adapter interface writes minimal information messages. The adapter writes trace message to the adapter_instance_name_trace.txt file in the LINK_DIR\adapters\logs directory.

Option	Description
Additional Java Launcher Options	Enables when launching the Java process that hosts the adapter. ℹ Note If you are connecting to the adapter from behind a proxy server, append the following to the Additional Java Launcher options: `-Dhttp.proxyHost=<proxy_server_name> -Dhttp.proxyPort=<proxy_server_port>`
Adapter type name	(Read-only) The name of the adapter used to create this instance.
Adapter version	(Read-only) The version of the adapter used to create this instance.
Adapter Class	(Read-only) A name that identifies the adapter class. The name depends on the type of adapter.

6. Click *Apply* and the Administrator adds your adapter instance to the list of those available to the SAP Data Services system.
7. Start the adapter, verify that it functions.

Related Information

Start and stop the adapter instance [page 2392]

11.3.2.2 Start and stop the adapter instance

Click the *Status* tab to view the status of all adapter instances you configured. From this tab, you can Start adapter instances and Shutdown or Abort instances that are running.

From the *Status* tab, you can also navigate to view Adapter Instance configuration details, Log Files, and Dependent Objects for each configured adapter instance.

11.3.3 Create the datastore

To associate the SAP Data Services Adapter for Salesforce.com with data flows, you must create an adapter datastore in the Designer. For general information on creating an adapter datastore, refer to the Datastores section of the *Designer Guide*.

11.3.3.1 To create an Adapter for Salesforce.com datastore

1. In the Datastores tab of the Designer Object Library, right-click and select *New*.

   The *Datastore Editor* window appears.
2. Type a unique, descriptive name in the *Datastore name* box.
3. Select *Adapter* from the *Datastore type* list.
4. Select the *Job server* associated with the adapter for Salesforce.com.
5. For *Adapter instance name*, choose the instance name you configured in the Administrator.
6. Click the *Advanced* button to access Adapter Options. Configure the following options:
   a) *Username* and *Password* (for Salesforce.com access)
   b) *Web service end point* (*https://www.salesforce.com/services/Soap/u/6.0*)
   c) *Batch size* (200 is default)
   d) *Metadata resilience?* Select *yes* if you want the adapter to perform in any of the following ways without throwing an error message (*no* is default)

When reading from normal or CDC sources	When reading from normal or CDC sources	When loading data to Salesforce.com
If a table no longer exists, the adapter sends no record of that table to SAP Data Services	If a table no longer exists, the adapter sends no record of that table to SAP Data Services.	If a table no longer exists, the adapter sends no data for that table to Salesforce.com.
If a field use in a data flow no longer exists, the adapter returns a NULL value for that field to Data Services	If a field used in a dataflow no longer exists, the adapter retains a NULL value for that field to Data Services.	If a column no longer exists, the adapter sends no value for that column to Salesforce.com.
If a field used in a WHERE clause no longer exists, all conditions that use that field automatically evaluate to FALSE, possibly reducing the conditions. For example, if the WHERE clause is 'WHERE ColumnA = A and (ColumnB = B or ColumnC = C)' and ColumnC no longer exists, the clause will be processed as follows: 'WHERE ColumnA = A and (ColumnB = B or ColumnC = C)' 'WHERE ColumnA = A and (ColumnB = B or FALSE)' 'WHERE ColumnA = A and ColumnB'		

The software can push the date and datetime fields down to Salesforce.com if you use the software's default formats ("yyyy-mm-dd hh:mi:ss" for datetime and "yyyy-mm-dd" for date) in your WHERE clause.

> Note
>
> Salesforce.com does not support the "like" operator on an ID type field. the software maps this type to varchar. The software's optimizer is unable to recognize a Salesforce.com ID field and cannot push down SQL statements containing the "like" operator in the ID field.

e) *Enable CDC* Select *yes* to configure as a CDC datastore (*no* is default).

> Note
>
> To avoid data processing problems, it is recommended that after you create the datastore you do not modify the Enable CDC value. Instead, create a new datastore and configure with the other Enable CDC value.

f) Set *Default Based64Binary field length* to establish the default length for a Salesforce.com Base64Binary field.

7. Click *OK* to save values and finish creating the datastore.

   If you did not provide the correct user name and password, or if you entered an invalid parameter, you will see an error message stating that the "Adapter connection failed."

11.3.4 Working with Salesforce.com metadata

The Salesforce.com adapter supports only tables (not function calls, documents, and so on).

11.3.4.1 Browse and import metadata

For general information on how to browse and import metadata using a SAP Data Services datastore, see the Datastores section of the *Designer Guide*.

11.3.4.1.1 To browse and import adapter metadata

1. Double-click your adapter datastore icon, or double-click the *Tables* icon underneath the adapter datastore icon. Alternatively, you can right-click the datastore icon and select *Open*.

   The Adapter Metadata Browser window opens with a list of table objects (and their descriptions) from Salesforce.com that are available for viewing.
2. Click to open nodes and browse the available metadata.

   Two or three folders appear under each table node. These folders include: Referenced by, References, and Columns.

   - The Referenced by and References folders show relationships between the expanded table and itself as well as other tables. (For example, if a Contact belongs to an Account, it will have an AccountId column pointing to its parent account. So, Account is "referenced by" Contact and Contact "references" Account.)

- The Columns folder lists the table columns and their descriptions.

3. Right-click any node to find out if that metadata can be imported into SAP Data Services. If *Import* appears as a right-click menu option, select it to import the metadata object.

Option	Imports
Table node	That table
Referenced By node	All tables directly under that node
References node	All tables directly under that node

4. You can also import metadata by name.
   a) Go to the Datastores tab of the object library.
   b) Right-click the adapter datastore and choose *Import By Name*.
   c) In the *Import By Name* window, enter the full, exact table name in the *Value* column.

11.3.4.2 The DI_PICKLIST_VALUES table

The Salesforce.com adapter includes a SAP Data Services proprietary table you can import like any other Salesforce.com table. This table contains all Salesforce.com picklists (a set of enumerated values from which to select). To use the DI_PICKLIST_VALUES table as a source in data flows, import the DI_PICKLIST_VALUES just like you would any other table, then drag-and-drop it as a source in your data flow. Connect to a Query transform and drill down to add a WHERE clause and filter the values you require. Columns defined for this table include:

OBJECT_NAME, FIELD_NAME, VALUE, IS_DEFAULT_VALUE, IS_ACTIVE, and LABEL.

i Note

If you have translated pickup values in Salesforce.com, the LABEL column returns values for the language specified in your personal information settings. If pickup values are not translated, the VALUE and LABEL columns return the same values.

11.3.4.3 Open and delete imported metadata

You can open imported metadata to view input and output schemas. To open an imported table, double-click its icon. To find the icon go to the adapter datastore in the object library and open *Tables*.

From the Designer, you can also delete imported metadata by right-clicking an imported object and selecting *Delete* from the menu.

After you import metadata, it is available for use in Data Services data flows.

11.3.4.4 Metadata mapping

Salesforce.com data types map to SAP Data Services data types as follows:

Salesforce Datatype	Description	SAP Data Services Datatype
xsd:base64Binary	Base 64-encoded binary data	varchar
xsd:boolean	Boolean (True/False) values	varchar ('true' or 'false')
xsd:date	Date values	date
xsd:datetime	Date/time values (time-stamps)	datetime
xsd:double	Double values	decimal
xsd:int	Integer values	int
xsd:string	Character strings	varchar

The date/time values that the Salesforce.com adapter retrieves from Salesforce.com are all in ISO 8601 format, reflect GMT time, and include a time zone field. To adjust for any time zone differences, the Salesforce.com adapter automatically performs a translation based on the associated local and server clocks. When the Salesforce.com adapter communicates datetime information to SAP Data Services, the software receives those values in local time and the time zone field is not considered.

> **Note**
>
> If your local and server clocks are not synchronized, translation speed is unaffected. However, if your local clock is not set to the correct time, the software may send incorrect times to Salesforce.com and changes that you expected to be returned may not be returned until a later synchronization.

Examples:

- If we are in Pacific Standard Time (PST) and the adapter receives '2005-08-10T23:00:00Z' (where 'Z' means GMT time) from Salesforce.com, the value sent to the software will be '2005.08.10 15:00:00'.
- You want to retrieve information that has changed since yesterday at 6:00 PM local time. You write a condition stating: SFDC_TIMESTAMP >='2005.08.10 18:00:00' and the software sends this condition "as is" to the adapter. Because Salesforce.com will not understand this timestamp (it lacks a time zone indicator), the Salesforce.com adapter automatically converts the time specified in the software to a format that Salesforce.com understands, formatting the value to '2005-08-11T01:00:00Z'.

11.3.4.5 CDC datastore tables and generated columns

The CDC table nodes differ from normal tables. If you expand a CDC table node, you will only see a Columns folder that contains the same columns as the original table with three generated columns. The generated columns are used for CDC data retrieval. Generated columns include:

- DI_SEQUENCE_NUMBER: The sequence number (int).
- DI_OPERATION_TYPE: The operation type (varchar).
- SFDC_TIMESTAMP: The Salesforce.com timestamp (datetime).

11.3.5 Design flows

After importing metadata as datastore objects in the Designer, you can use that metadata when designing data flows.

(For general application design and administration information, see the *SAP Data Services Designer Guide* and the *SAP Data Services Administrator Guide*.)

11.3.5.1 Changed data and Salesforce.com

One simple usage of the Salesforce.com tables is to read changed data. The following example explains one way you can schedule SAP Data Services to query Salesforce.com for changed data after loading Salesforce.com tables into your local repository.

11.3.5.1.1 Reading changed data from Salesforce.com

1. Import CDC table metadata into your local repository.
2. Build a data flow by selecting a CDC table as a source object and connecting that source to a Query transform.
3. Drill into the source object and select the following tabs to set CDC-related options:

   CDC Options. CDC table options include:

Option Name	Description
CDC subscription name	(Required) A name that Data Services uses to keep track of your location in a continuously growing Salesforce.com CDC table. Salesforce.com CDC uses the subscription name to mark the last row read so that the next job starts reading the CDC table from that position. You can use multiple subscription names to identify different users who read from the same imported Salesforce.com CDC table. Salesforce.com CDC uses the subscription name to save the position of each user. Type a new name to create a new subscription. A subscription name must be unique within a datastore, owner, and table name. For example, you can use the same subscription name without conflict with different tables that have the same name in the same datastore if they have different owner names. The software requires that you enter a value for this option.
Enable check-point	Enables the software to restrict CDC reads using check-points. After a check-point is in place, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last check-point. By default, check-points are not enabled.

Option Name	Description
Get before-image for each update row	Some databases allow two images to be associated with an UPDATE row: a before-image and an after-image. If your source can log before-images and you want to read them during change-data capture jobs, enable this option. By default, the software retrieves only after-images.

a) Specify a value for the *CDC subscription name*.

b) If you select *Enable check-point*, the software remembers the timestamp of last load and automatically applies that timestamp as the start time for the next load. By using the *Enable check-point option*, you do not need to define a WHERE clause in the Query transform.

c) Do not select *Get before-image for each update row* (for use only if your source can log before-images and you want to read them during change-data capture jobs) as Salesforce.com provides no before-images.

Adapter source options include:

Option Name	Description
Column delimiter	Specify a one-character delimiter for data columns by entering the forward-slash (/) followed by a three-digit ASCII code to indicate an invisible character.
Row delimiter	Specify a one-character delimiter for data rows by entering the forward-slash (/) followed by a three-digit ASCII code to indicate an invisible character.
Escape character	Must be one character.
CDC table source default start date	This option works with the CDC *Enable check-point* option. Salesforce.com requires the software to supply a start date and end date as part of a changed data request.
Fetch deleted records	Set this value to *Yes* to also fetch the deleted records from the table. The default value is *No*.

4. Add a Map_CDC_Operation transform after the Query transform.
5. Drill into the Map_CDC_Operation transform and configure the CDC columns in the transform editor.
   - Note that the software automatically pre-populates the *Sequencing column* and the *Row operation columns* fields with DI_SEQUENCE_NUMBER and DI_OPERATION_TYPE, respectively.
     The software fills DI_SEQUENCE_NUMBER using sequential numbers starting at 0 every time the CDC operation starts. Returned rows are always sorted by this column.
     The DI_OPERATION_TYPE indicates the type of operation performed on the object: INSERT, UPDATE or DELETE (I, U or D). The adapter does not return before-image records (B).
   - The SFDC_TIMESTAMP value will always indicate the time at which the operation was performed, (when the object was inserted, deleted, or last updated).
   - The other column values may or may not be set by the software, depending on the operation type. For a DELETE operation, only the ID will be set. For UPDATE and INSERT, the columns are set to represent the state of the object after the operation.
6. Connect the Map_CDC_Operation transform to your target table (where the INSERT, UPDATE and DELETE commands will be executed).

The following table shows the CDC operation mapping of data from Salesforce.com to the software:

Salesforce.com data since last CDC operation	Records returned to Data Services
INSERT	INSERT

Salesforce.com data since last CDC operation	Records returned to Data Services
UPDATE	UPDATE
DELETE	DELETE
INSERT & UPDATE	INSERT & UPDATE
INSERT & DELETE	DELETE
UPDATE & DELETE	DELETE
INSERT & UPDATE & DELETE	DELETE

If an object was inserted and updated after the reference time, two CDC records are returned to the software, one for each operation. However, both records will contain the same information, reflecting the state of the object after the UPDATE. So, in this type of situation, there is no way of knowing the object state after the INSERT operation.

Related Information

Designer Guide: Techniques for Capturing Changed Data, Using mainframe check-points [page 758]
Designer Guide: Techniques for Capturing Changed Data, Using before images from mainframe sources [page 759]

11.3.5.1.2 Using check-points

If you can replicate an object, Salesforce.com allows applications to retrieve the changed data for that object. Salesforce.com saves changed data for a limited amount of time (for details, see your Salesforce.com technical documentation). Salesforce.com monitors neither the retrieving application nor the data retrieved.

When you enable check-points, a CDC job in Data Services uses the subscription name to read the most recent set of appended rows and to mark the end of the read (using the SF_Timestamp of the last record). If you disable check-points, the CDC job always reads all the rows in the CDC data source which increases processing time.

To use check-points, on the Source Table Editor enter the CDC Subscription name and select the *Enable check-point* option. If you enable check-points and run a CDC job in recovery mode, the recovered job begins to review the CDC data source at the last check-point.

Note

To avoid data corruption problems, do not reuse data flows that use CDC datastores because each time a source table extracts data it uses the same subscription name. This means that identical jobs, depending upon when they run, can get different results and leave check-points in different locations in the file.

11.3.5.1.3 Using the CDC table source default start date

The CDC table source default start date is dependent on several factors. This date can be a value you specify, a check-point value, or a date related to the Salesforce.com retention period.

When you do not specify a value for the start date:

- SAP Data Services uses the beginning of the Salesforce.com retention period as the start date if a check-point is not available (during initial execution).
- The software uses the check-point as the start date if a check-point is available and occurs within the Salesforce.com retention period. If the check-point occurs before the retention period, the software uses the beginning of retention period as the start date.
- However, if a table is created within the Salesforce.com retention period and a check-point is not available, the execution returns an error message. Drill into the source object and enter a value for the CDC table source default start date. The value must be a date that occurs after the date the table was created to work around this problem.

When you specify a start date value, if your date occurs:

- Within the Salesforce.com retention period and no check-point is available, then the software uses your specified value.
- Within the Salesforce.com retention period and after the check-point, the software uses your specified value.
- Within the Salesforce.com retention period and before the check-point, the software uses the check-point value as the start date.
- Outside of the Salesforce.com retention period, the Salesforce.com Adapter ignores the value.

11.3.5.1.4 Limitations

The Table Comparison and SQL transforms and the lookup and lookup_ext functions cannot be used with a source table imported with a CDC datastore because of the existence of the SAP Data Services generated columns. You cannot compare or search these columns.

11.3.6 Run applications

After you design your application(s), you must run them to finalize SAP Data Services-Salesforce.com integration. These are the basic startup tasks:

- In the Administrator, start each application to be used in the integration.
  Real-time: Start services and applications that use this service.
  Batch: Start/schedule the job.
- In the Administrator, monitor progress for each job. You can monitor pending requests, processed requests, failed requests, and status.

> Note
>
> The Administrator does not automatically refresh views. To refresh views, go to the View menu and select Refresh.

- In the Administrator, monitor progress for each (real-time) service.
- On the Salesforce.com Server, monitor messaging progress for the configured queues.

If problems occur:

- For error message descriptions and suggested troubleshooting actions, see the Understanding error messages section.
- To understand the source of a problem, use error and log tracing.
- To enable debug tracing for the adapter instance, use the Administrator.

Related Information

Understanding error messages [page 2401]

11.3.7 Understanding error messages

During the course of designing and deploying your jobs, you may encounter error messages. Find error messages and their descriptions (including suggested actions) listed in the following table:

Error Message	Description
Login operation has failed. SForce.com message is {0}	Invalid user name/password or user account is blocked for another reason, which is explained by the Salesforce.com message. ACTION: Confirm password or contact Salesforce.com for more information.
Unknown object type. SForce.com message is {0}	The table used in the query is no longer available or visible to the user. ACTION: Browse Salesforce.com metadata and look for the table.
Invalid field. SForce.com message is {0}	One or more fields used in the query are no longer available. ACTION: Browse Salesforce.com metadata to determine if there is a difference between the imported table and the actual metadata. If necessary, rebuild your data flow.
Unsupported SQL statement: {0}	Your data flow is not supported by Salesforce.com. ACTION: Rebuild according to the restrictions described in this document.
Malformed query: {0}. SForce.com message is {1}	The submitted query is unsupported by Salesforce.com. Most likely you have encountered a bug translating between data flows and Salesforce.com queries. ACTION: Contact product support.
Invalid session parameter: name = {0}, value = {1}	The URL or batchSize session parameter is invalid. Either the URL is malformed or batchSize is not a positive integer.

Error Message	Description
	ACTION: Check the integrity of the URL and confirm that the batchSize is a positive integer.
Invalid CDC query: {0}	The data flow built over a CDC table is invalid. ACTION: Check for (and fix) any missing WHERE clause condition for SFDC_TIMESTAMP.
There was a service connection error when talking to SForce.com: {0}	The adapter could not connect to Salesforce.com. ACTION: Confirm that the web service end point is correct and accessible through your network.
There was a communication error when talking to SForce.com: {0}	A protocol error occurred. ACTION: Contact product support.
There was an unexpected error. SForce.com message is {0}	An unknown, unexpected error occurred. ACTION: Contact product support.

12 Supplement for SAP

12.1 Introduction

This guide contains information about how to use Data Services with SAP applications and SAP NetWeaver Business Warehouse. It is organized to complement the information in the *Data Services Designer Guide*, and the reference section complements the information in the *Data Services Reference Guide*.

12.1.1 Terms

The following table describes SAP terminology for this release.

Term	Abbreviation	Description
SAP Applications	Not applicable	Applications are the main building blocks of the SAP solution portfolios for industries. They provide the software foundation with which organizations address their business issues. SAP delivers the following types of applications: • General-purpose applications: These include applications provided within SAP Business Suite software such as the SAP Customer Relationship Management application and the SAP ERP application. • Industry-specific applications: These applications perform targeted, industry-specific business functions. Examples are the SAP Apparel and Footwear application for the consumer products industry and the SAP Reinsurance Management application for the insurance industry. SAP R/3 software is available today as SAP ERP. Therefore, SAP R/3 software functionality falls under the category of SAP Applications.
SAP Master Data Services	Not applicable	A solution powered by SAP HANA that provides enterprises with instant access to a complete view of their entire customer’s information and

Term	Abbreviation	Description
		relationships—data that is seamlessly consolidated and enriched from heterogeneous sources across the enterprise.
SAP NetWeaver technology platform	Not applicable	Helps organizations perform a wide variety of IT processes such as user productivity enablement, information management, and business process management. Includes several components and tools
SAP NetWeaver Application Server component	SAP NetWeaver AS	Supports platform-independent Web services, business applications, and standards-based development. This software enables you to leverage existing technology assets for Web-services-oriented solutions. It is a component of the SAP NetWeaver technology platform. Formerly referred to as the SAP Web Application Server or Basis.
SAP NetWeaver Business Warehouse component	SAP NetWeaver Business Warehouse	Enables you to integrate data from across the enterprise and transform it into practical, timely business information to drive sound decision-making. It is a component of the SAP NetWeaver technology platform.

12.1.2 SAP interfaces

Data Services includes the following SAP interfaces:

- SAP Applications ODP [Operational Data Provisioning] sources
- SAP Applications ABAP
- SAP Applications BAPI (Business Application Programming Interface)
- SAP Applications IDoc
- SAP Master Data Services
- SAP NetWeaver Business Warehouse (BW)
- SAP NetWeaver BW Open Hub Destination service

The following table lists SAP interfaces, their intended functionality, and the types of data flows you can use with each.

Feature	How is this feature designed to be used?	Which data flow should be used?
SAP applications ODP sources i Note Refer to the *What's New Guide* for ODP sources	Use an SAP Applications datastore to import metadata from SAP ODP sources to create batch jobs that extract data from SAP application sources.	data flow or ABAP data flow
SAP Applications ABAP	An alternative to SAP ODP sources. Use an SAP Applications datastore to import metadata (tables, files, hierarchies, and functions) to create batch jobs to extract data from SAP application sources.	ABAP data flow
SAP Applications BAPI	Use an SAP Applications datastore to import BAPI function metadata to be included in batch or real-time jobs. BAPI functions can be used to update SAP application and SAP NetWeaver BW sources.	data flow
SAP Applications IDoc	Use an SAP Applications datastore to import IDoc metadata which can be used to create the following for batch and real-time jobs: For batch jobs • IDoc file sources • IDoc message sources • IDoc message targets For real-time jobs • IDoc file sources • IDoc message sources • IDoc file targets • IDoc message targets	data flow
SAP Master Data Services	Use an SAP Master Data Services datastore to import SAP Master Data Services metadata to create batch jobs that can be run from Data Services to read from SAP Master Data Services sources or update SAP Master Data Services targets.	data flow
SAP NetWeaver BW	Use an SAP BW Target datastore to import BW metadata to create batch jobs that can be run from SAP NetWeaver BW or Data Services to update SAP BW targets.	data flow (batch jobs only)
SAP NetWeaver BW Open Hub Destination service	Use an SAP BW Source datastore to import Open Hub Destination tables that are sources from which Data Services can read data.	data flow

12.1.3 Version checking

Data Services provides version information for SAP application ABAP programs and functions. With this feature, Data Services can:

- Embed version information in ABAP programs (programs that are both provided by Data Services for an SAP application environment and generated by Data Services from ABAP data flows)
- Generate warnings to update ABAP programs that were uploaded during a previous Data Services installation
- Compare an ABAP program with its ABAP data flow and produce a warning if it finds a version mismatch
- Compare functions on an SAP application server (uploaded during a previous Data Services installation) with those required by the current Data Services installation and produce a warning if it finds a version mismatch

To avoid version mismatches, it is recommended that when you upgrade Data Services, use it with the transport files supplied in the same product package and regenerate ABAP programs from ABAP data flows if you pre-loaded them to an SAP application server.

12.1.3.1 Generated ABAP programs

If you select *Execute Preloaded* for the *ABAP execution option mode* in your SAP Applications datastore, when Data Services executes a job that contains an ABAP data flow, it runs the previously generated and uploaded ABAP program. If this program was generated by an old version of Data Services, the job might not be taking advantage of the performance improvements in the latest version.

You can set Data Services to trigger version warning messages when a job executes. If you find a version mismatch, upgrade the generated ABAP program on the SAP server to match the Data Services version.

12.1.3.2 To upgrade a generated ABAP program stored on an SAP application server

1. Open a project containing the ABAP data flow that you want to update.
2. In the project area, select the ABAP data flow.
3. Select *Validation* > *Generate ABAP code* .
4. Modify your ABAP program on the SAP application server using this new code.

> i Note
>
> If you select the *Generate and Execute* mode for an SAP Applications datastore configuration, Data Services generates an ABAP program each time a job runs. Data Services does not need to check an ABAP program's version if the program is generated each time a job runs.

12.1.3.3 Data Services ABAP programs

When you connect to an SAP application environment, Data Services makes use of and tracks version information for built-in ABAP programs. If you encounter a warning, upgrade the Data Services ABAP programs to match the current Data Services version.

12.1.3.4 Data Services functions

If you want to connect to an SAP application environment, Data Services provides functions for you to load onto an SAP application server. Data Services tracks version information for these functions and can trigger warning messages if it finds that functions from a previous Data Services installation exist on the server. If you encounter a warning, upgrade the Data Services functions on the SAP application server to match the current Data Services version.

12.1.3.5 Warning messages

You can enable and disable version checking.

12.1.3.6 To enable or disable version checking for ABAP programs and functions

1. From the Designer, select *Tools* > *Options* > *Job Server* > *General* .

   This dialog allows you to modify the configuration of the default Job Server.
2. Set the following keys in the al_engine section as needed.

   These keys control different SAP application warning messages:

Key	Messages
ABAP_PRELOAD_VERSION_MISMATCH	151201, 151203
ABAP_PRELOAD_KEY_MISMATCH	151202, 151204
ABAP_SUPPLIED_VERSION_MISMATCH	151205
SAP_FUNCTION_MISMATCH	151206

   The default for ABAP_PRELOAD_VERSION_MISMATCH and ABAP_PRELOAD_KEY_MISMATCH is FALSE, and the default value for ABAP_SUPPLIED_VERSION_MISMATCH and SAP_FUNCTION_MISMATCH is TRUE.

As described in the following sample scenarios, if you enable warnings for the ABAP and function version checking feature, Data Services provides an appropriate response:

Scenario	Response
After installing a new release of Data Services and enabling the version warning messages feature, you try to use Data Services against function metadata imported by an older Data Services version.	Data Services generates a warning message: 151206 3 3 4 "Warning: a mismatch was detected for SAP function <%s (%s)> : installed version <%s>. The expected version is <%s>. Please upload the correct function."
You just upgraded to a new version of Data Services but chose not to upgrade the SAP application server and you do not want to encounter warning messages.	You do nothing. However, it is recommended that you avoid this scenario by regenerating and uploading ABAP files when you upgrade Data Services.
You generate ABAP and upload it to your SAP server. After enabling all version warning messages and running it in execute_preloaded mode for a while, you change an ABAP data flow.	Data Services generates a warning message.
You generated an ABAP program and uploaded it to your SAP server. After enabling all version warning messages and running it in Execute Preloaded mode for a while, you upgrade Data Services.	Data Services generates a warning message because of the potential benefits you will gain by re-generating ABAP.

12.1.3.7 /BODS/GET_VERSION function

This function is included in each transport file for SAP applications.

/BODS/GET_VERSION retrieves the list of Data Services provided functions and their versions from the SAP application server.

I / P_FUNC / RFCFUNC-FUNCNAME

E / O_DI_VER / RFCFUNC-FUNCNAME

T / ENTRIES / TAB512

12.2 Installing Functions on the SAP Server

SAP Data Services provides functions that support the use of the ABAP, BAPI, and IDoc interfaces on SAP servers. You will need some or all of these functions in the normal operation of the software in an SAP environment. These functions perform such operations as dynamically loading and executing ABAP programs from Data Services, efficiently running preloaded ABAP programs, allowing for seamless metadata browsing and importing from SAP servers, and reporting the status of running jobs. Some of these functions read data from SAP NetWeaver BW sources.

You must upload the provided functions to your SAP server in a production environment. It is recommended that you always upload the functions to your SAP server whether you are in a development, test, or production environment. The functions provide seamless integration between Data Services and SAP servers.

The default installation places two function module files for SAP servers in the `...\Data Services\Admin\R3_Functions\transport` directory. You then upload these files to SAP servers using the SAP Correction and Transport System (CTS) or manually. Using CTS allows for version control as the functions evolve across releases.

The installation provides two versions of transport files (depending on the server version you are using) to install the functions on the SAP server. To obtain the names of the latest transport files for installing or upgrading these SAP server functions, see the readme.txt file in the `...\Data Services\admin\R3_Functions\transport` directory.

12.2.1 Installing the functions using CTS

If you are installing SAP Data Services for the first time or upgrading to a new release, you must add the provided functions to the SAP server. The recommended method is to use the SAP Change and Transport System (CTS). This procedure assumes you are familiar with CTS and have the authorization to perform the transport.

To install provided functions using the CTS system:

1. Copy the provided transport files to the appropriate directories on your SAP server.

   > **Note**
   >
   > In these steps, **`<900XXX.XXX>`** is a variable. To substitute the correct file name for the current release, see the readme.txt file in the `...\Data Services\admin\R3_Functions\transport` directory.

   a) Copy **`<R900XXX.XXX>`** to the `/usr/sap/trans/data` directory.
   b) Copy **`<K900XXX.XXX>`** to the `/usr/sap/trans/cofiles` directory.
2. Log on to the SAP server and run the transaction `/nSE37` to determine if function group /BODS/BODS already exists. The CTS system will install the Data Services functions into a single function group /BODS/BODS that it automatically creates on the SAP server if it does not already exist.

   If function group /BODS/BODS exists and contains previously installed Data Services functions, add an unconditional mode 2 option (parameter U2) to the `tp import` command described in step 4 below (`tp import <xxx>K900<xxx> <SID> U2`).
3. From a command window, run the following transport command:

   ```
   tp addtobuffer <xxx>K900<xxx> <SID>
   ```

   (where SID is the SAP system ID)

   You receive the response:

   ```
   This is tp version <SAP TP and SAP versions> for
               <database type> database. Addtobuffer successful for
               <xxx>K900<xxx> tp finished with a return code: 0 meaning:
               Everything OK
   ```
4. Run the next transport command:

   ```
   tp import <xxx>K900<xxx> <SID>
   ```

   (where SID is the SAP System ID)

   You receive a response that indicates the SAP TP and SAP versions, database type, SAP server name, and the TP return code, which indicates whether or not the TP command successfully uploaded the functions.

5. Run `/nSE37` again to verify that the functions uploaded.

When importing the functions, you might see an error such as the following:

```
1 of 1 requests do not match the component version of the target system
```

To prevent this error, in STMS enable the *Ignore Invalid Component Version* option.

12.2.2 Function modules

The names of the SAP Data Services functions have the prefix /BODS/ added to the corresponding SAP function names. The appropriate Data Services function modules for your SAP version are installed when you upload them to your SAP server.

12.2.2.1 Packaging of /BODS/ RFC_ABAP_INSTALL_AND_RUN

For SAP servers, SAP Data Services provides a function called /BODS/RFC_ABAP_INSTALL_AND_RUN in its transport file. This function is used in the generate_and_execute mode (set in an SAP datastore) to generate ABAP code on Data Services and execute it on an SAP server. This function allows dynamic program generation and is particularly useful in a development environment. However, it also provides a level of write access to an SAP server that might not be acceptable to your security environment. If this is a concern, remove this function from your SAP server.

12.2.2.2 Sample function module

The following example illustrates the function module /BODS/SYNTAX_CHECK:

Example

```
FUNCTION /BODS/SYNTAX_CHECK.
*"--------------------------------------------
*"*"Local interface:
*"    EXPORTING

    *"         VALUE(ERRORMESSAGE) LIKE SY-MSGV1
*"    TABLES
  *"         PROGRAM STRUCTURE PROGTAB OPTIONAL
*"--------------------------------------------
DATA: MESS(72), LIN(72), WRD(72).
   SYNTAX-CHECK FOR PROGRAM MESSAGE MESS LINE LIN WORD WRD.
  IF MESS <> SPACE.
  ERRORMESSAGE = MESS.
  EXIT.
   ENDIF.
ENDFUNCTION.
```

12.2.2.3 Function parameters

The following table describes information about functions, their interfaces, and function source code. This table describes the parameters with the following format:

```
Parameter type / Parameter name / Reference field
```

The parameter types are:

- I: Import parameter
- E: Export parameter
- T: Table parameter

Function name	Description

/BODS/ABAP_RUN	Executes generated or preloaded ABAP programs.
	`I / PROGRAMNAME / SY-REPID`
	`E / ERRORMESSAGE /  SY-MSGV1`
	`T / SELTAB /  RSPARAMS`
	`LOG /  LISTZEILE`
	Exception: PROGRAM_DOES_NOT_EXIST

/BODS/AUTH_IMPORT	Imports SAP application authorization profiles used by /BODS/ABAP_RUN.
	`I / AUTHPROF /  USR10-PROFN`
	`T / AUTHOBJ /  LISTZEILE`
	Exception: SECURITY_PROFILE_DOESNOT_EXIST

/BODS/BW_QUERY	Gets metadata information from an SAP BW system.
	`I / P_CODE / DD01D-DATATYPE`
	`I / P_IAREA / DD01D-DDTEXT`
	`I / P_ICUBE / DD01D-DDTEXT`
	`I / P_ODSO / DD01D-DDTEXT`
	`T/ ENTRIES / TAB512`

/BODS/COLUMN_SEARCH	/BODS/COLUMN_SEARCH is currently not used and is reserved for future implementation.
	`I / P_TNAME / DD03VT-DDTEXT`
	`I / P_CNAME / DD03VT-DDTEXT`
	`I / P_CDESC / DD03VT-DDTEXT`
	`I / P_LANG / DD03VT-DDLANGUAGE`
	`I / P_MAXROWS / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

/BODS/DATA_PROFILE	Profiles an SAP table.
	` I / P_CODE / DD02L-TABNAME`
	` I / P_TABLE / DD02L-TABNAME`
	`I / P_FIELD / DD03L-FIELDNAME`
	`T / WRITES / LISTZEILE`

Function name	Description

/BODS/EXTRACTOR_IMPORT	Allows import of an ERP extractor metadata into the Data Services repository.
	`I / P_LANG / DD02T-DDLANGUAGE DEFAULT SY-LANGU`
	`I / P_NAME / ROOSOURCE-OLTPSOURCE`
	`E / P_ERRORMESSAGE / SY-MSGV1`
	`T / P_RESULT / LISTZEILE`
	`T / P_VIEW_DEF / LISTZEILE`
	Exceptions:
	● EXTRACTOR_IS_NOT_SUPPORTED
	● NO_AUTHORITY
	● GET_METADATA_FAIL
	● EXTRACTOR_DOES_NOT_EXIST

/BODS/EXTRACTOR_NAVIGATE	Allows various ways of listing available ERP extractors in an SAP system.
	`I / P_LANG / DD02T-DDLANGUAGE DEFAULT SY-LANGU`
	`I / P_APPLNM / ROOSOURCE-APPLNM`
	`I / P_TYPE / ROOSOURCE-TYPE`
	`I / P_ALLEXT / CHAR_01`
	`T / P_RETURN / LISTZEILE`

/BODS/EXTRACTOR_SEARCH	Allows searches for ERP extractors.
	`I / P_LANG / DD02T-DDLANGUAGE DEFAULT SY-LANGU`
	`I / P_NAME / ROOSOURCE-OLTPSOURCE`
	`I / P_MAX_ROWS / I`
	`I / P_FLAG / I`
	`T / P_RETURN / LISTZEILE`

/BODS/FILE_ROWCOUNT	Puts/gets the row count information for each transported file in the data flow into/from the /BODS/BODS table.
	`I / NAME / /BODS/BODS-NAME`
	`I / ROWCOUNT / /BODS/BODS-TOTAL_ROW`
	`I / SET / SONV-FLAG`
	`E / TOTAL_ROW / /BODS/BODS-TOTAL_ROW`

/BODS/FUNCTION_GET	Gets the function interface.
	`I / FUNCNAME / TFDIR-FUNCNAME`
	`T / PRMTAB / CATFU`

/BODS/GET_VERSION	Retrieves the list of Data Services provided functions and their versions from the SAP application server. This function is included in each transport file.
	`I / P_FUNC / RFCFUNC-FUNCNAME`
	`E / O_DI_VER / RFCFUNC-FUNCNAME`
	`T / ENTRIES / TAB512`

Function name	Description

/BODS/IDOC_IMPORT	Imports IDocs from SAP applications.
	`I / IDOCTYP / EDISYN-DOCTYP`
	`I / LANG /  EDISEGT-LANGUA`
	`T / SEGMENTS / LISTZEILE`
	Exception: IDOC_TYPE_DOESNOT_EXIST

/BODS/IDOC_SEARCH	Searches existing IDocs in SAP applications.
	`I / P_NAME / EDIDOT-DOCTYP`
	`I / P_DESC /  EDIDOT-DESCRP`
	`I / P_LANG / EDIDOT-LANGUA`
	`I / P_MAXROWS / SY-TABIX`
	`I / P_FLAG / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

/BODS/JOB_LOG	Provides error handling and retrieves the SAP application job log when a job is cancelled. This function is required for the execute_preloaded access method you can define for an SAP Applications datastore.
	`I / JOBCOUNT / TBTCO-JOBCOUNT`
	`I / JOBNAME / TBTCO-JOBNAME`
	`I / HOST / TBTCO-BTCSYSREAX`
	`T / LOG / BTCTLE`
	Exceptions:
	• JOB_NOT_FOUND
	• JOB_LOG_NOT_FOUND
	• JOB_NUMBER_NOT_VALID
	• LOGNAME_WRONG_FORMAT
	• LOGNAME_MISSINGLOG_HANDLE_ERROR
	• LOG_IS_EMPTY
	• LOG_NOT_FOUND
	• TEMSE_CONVERSION_NOT_POSSIBLE
	• TEMSE_ERROR
	• TEMSE_FUNCTION_ERROR

/BODS/JOB_RUN	Submits ABAP jobs for background execution.
	`I / PROGRAMNAME / SY-REPID`
	`I / JOBGROUP /  TBTCO-JOBGROUP`
	`I / JOBNAME / TBTCO-JOBNAME`
	`I / HOST / TBTCO-BTCSYSREAX`
	`I / JOBCLASS / TBTCO-JOBCLASS`
	`E / JOBCOUNT /  TBTCO-JOBCOUNT`
	`E / JOB_RELEASED / BTCH0000-CHAR1`
	`T / SELTAB /  RSPARAMS`
	Exceptions:
	• ABAP_PROGRAM_SYNTAX_ERROR

Function name	Description
	• CANT_CREATE_JOB • INVALID_JOB_DATA • JOBNAME_MISSING • CANT_START_IMMEDIATE • INVALID_STARTDATE • JOB_CLOSE_FAILED • JOB_NOSTEPS • JOB_NOTEX • LOCK_FAILED • ABAP_PROGRAM_DOES_NOT_EXIST

/BODS/ JOB_STATUS	Queries ABAP job status after a job is successfully submitted.
	`I / JOBNAME / TBTCO-JOBNAME`
	`I / JOBCOUNT / TBTCO-JOBCOUNT`
	`I / HOST / TBTCO-BTCSYSREAX`
	`E / STATUS / TBTCO-STATUS`
	Exception: JOB_NOT_FOUND

/BODS/ MODEL_NAVIGATE	Allows you to navigate through the SAP data model, helping to identify tables.
	`I / P_OID / DM41S-DMOID`
	`I / P_LANG / DD02T-DDLANGUAGE`
	`T / P_RETURN / LISTZEILE`

/BODS/ READ_TEXT	Allows extraction of text from SAP applications. This function is an RFC-enabled version of the SAP application function module, READ_TEXT.
	`I / CLIENT / SY-MANDT`
	`I / ID / THEAD-TDID`
	`I / LANGUAGE / THEAD-TDSPRAS`
	`I / NAME / THEAD-TDNAME`
	`I / OBJECT / THEAD-TDOBJECT`
	`I / ARCHIVE_HANDLE / SY-TABIX`
	`E / HEADER / THEAD`
	`T / LINES / TLINE`
	Exceptions:
	• ID: Text ID invalid
	• LANGUAGE: Invalid language
	• NAME: Invalid text name
	• NOT_FOUND: Text not found
	• OBJECT: Invalid text object
	• REFERENCE_CHECK: Reference chain interrupted
	• WRONG_ACCESS_TO_ARCHIVE: Archive handle invalid

Function name	Description

/BODS/RFC_ABAP_INSTALL_AND_RUN	Allows Data Services to dynamically generate and run programs on an SAP server.
	`I / MODE / SY-MSGTY`
	`I / PROGRAMNAME / SY-REPID`
	`E / ERRORMESSAGE / SY-MSGV1`
	`T / PROGRAM / PROGTAB`
	`T / WRITES / LISTZEILE`

/BODS/RFC_READ_EXTRACTOR	Allows extraction of data from an ERP extractor.
	`I / P_NAME / ROOSOURCE-OLTPSOURCE`
	`I / P_DELIMITER / SONV-FLAG`
	`I / P_ROWCOUNT / SOID-ACCNT`
	`I / P_IS_VIEWDATA / CHAR_01`
	`I / P_WHERE /  /BODS/CH2K-WA`
	`I / P_SIZE / RSIDOCSIZE`
	`E / P_OUT_TABLE / DD02L-TABNAME`
	`T / P_FIELDS / RFC_DB_FLD`
	`T / P_TBLOUT128 / /BODS/CH128`
	`T / P_TBLOUT512 / /BODS/CH512`
	`T / P_TBLOUT2048 / /BODS/CH2K`
	`T / P_TBLOUT8192 / /BODS/CH8192`
	`T / P_TBLOUT30000 / /BODS/CH30K`
	Exceptions:
	• DATA_BUFFER_EXCEEDED
	• EXTRACTOR_INIT_FAIL
	• EXTRACTOR_GETNEXT_FAIL
	• NO_AUTHORITY
	• GET_METADATA_FAIL
	• SYNTAX_ERROR
	• FIELD_NOT_VALID

/BODS/RFC_READ_TABLE	Allows extraction of SAP application table data.
	`I / QUERY_TABLE / DD02L-TABNAME`
	`I / DELIMITER / SONV-FLAG DEFAULT SPACE`
	`I / NO_DATA / SONV-FLAG DEFAULT SPACE`
	`I / ROWSKIPS / SOID-ACCNT DEFAULT 0`
	`I / ROWCOUNT / SOID-ACCNT DEFAULT 0`
	`T / OPTIONS / RFC_DB_OPT`
	`T / FIELDS / RFC_DB_FLD`
	`T / DATA / /BODS/CH2K`
	Exceptions:
	• TABLE_NOT_AVAILABLE
	• TABLE_WITHOUT_DATA
	• OPTION_NOT_VALID
	• FIELD_NOT_VALID
	• NOT_AUTHORIZED
	• DATA_BUFFER_EXCEEDED

Function name	Description

/BODS/RFC_READ_TABLE2	Allows extraction of SAP application table data (enhanced version of /BODS/RFC_READ_TABLE).
	`I / QUERY_TABLE / DD02L-TABNAME`
	`I / DELIMITER / SONV-FLAG`
	`I / NO_DATA / SONV-FLAG`
	`I / ROWSKIPS / SOID-ACCNT`
	`I / ROWCOUNT / SOID-ACCNT`
	`E / P_OUT_TABLE / DD02L-TABNAME`
	`T / OPTION S/ RFC_DB_OPT`
	`T / FIELDS / RFC_DB_FLD`
	`T / TBLOUT128 / /BODS/TAB128`
	`T / TBLOUT512 / /BODS/TAB512`
	`T / TBLOUT2048 / /BODS/TAB2048`
	`T / TBLOUT8192 / /BODS/TAB8192`
	`T / TBLOUT30000 / /BODS/TAB30K`
	Exceptions:
	• TABLE_NOT_AVAILABLE
	• TABLE_WITHOUT_DATA
	• OPTION_NOT_VALID
	• FIELD_NOT_VALID
	• NOT_AUTHORIZED
	• DATA_BUFFER_EXCEEDED

/BODS/RFC_READ_TABLE_FILE	Allows extraction of SAP application table data (writes the data to the specified file).
	`I / QUERY_TABLE / DD02L-TABNAME`
	`I / DELIMITER / SONV-STATUS`
	`I / NO_DATA / SONV-FLAG`
	`I / OUT_DIR / / BODS/CH2K`
	`I / DATA_FILE / /BODS/CH2K`
	`I / EMPTY_FILE / SONV-FLAG`
	`T / OPTIONS / RFC_DB_OPT`
	`T / FIELDS / RFC_DB_FLD`
	Exceptions:
	• TABLE_NOT_AVAILABLE
	• TABLE_WITHOUT_DATA
	• FIELD_NOT_VALID
	• NOT_AUTHORIZED
	• DATA_BUFFER_EXCEEDED

/BODS/SYNTAX_CHECK	Performs a syntax check for generated or preloaded ABAP. This function is required for the execute_preloaded access method you configure in an SAP Applications datastore.
	`E / ERRORMESSAGE / SY-MSGV1`
	`T / PROGRAM /  PROGTAB`

/BODS/TABLE_IMPORT	Imports SAP application table definitions into the Data Services repository.
	`I / TABNAME / DD03L-TABNAME`
	`I / LANG / DD02T-DDLANGUAGE`

Function name	Description

	`E / DESCRIPTION / DD02T-DDTEXT`
	`E / TABCLASS / DD02L-TABCLASS`
	`T / COLUMNS / LISTZEILE`
	`T / INDEXES / LISTZEILE`
	`T / FKEYS / LISTZEILE`
	Exceptions: NO_TABLECLASS_FOR_TABLE

/BODS/TABLE_SEARCH	Allows searches for SAP application tables.
	`I / P_NAME / DD02-DDTEXT`
	`I / P_DESC / DD02VV-DDTEXT`
	`I / P_LANG / DD02V-DDLANGUAGE`
	`I / P_MAXROWS / SY-TABIX`
	`I / P_FLAG / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

/BODS/TEXTS	Allows extraction of texts for program areas (currently not used).
	`I /P_CODE/PROGTAB-LINE`
	`T /ENTRIES/TAB512`

/BODS/TREE_IMPORT	Allows you to import ERP hierarchy metadata into the Data Services repository.
	`I / P_LANG / DD04T-DDLANGUAGE`
	`I / P_TREEAREA / DD02D-DDTEXT`
	`I / P_TREEGROUP / DD04L-ROLLNAME`
	`I / P_TREETABLE / DD02D-TABNAME`
	`E / DESCRIPTION / DD04T-DDTEXT`
	`E / CONTROLLING_AREA / DD02D-DDTEXT`
	`E / CHART_OF_ACCT / DD02D-DDTEXT`
	`E / TABLE_NAME / DD02D-TABNAME`
	`E / FIELD_NAME / DD03D-FIELDNAME`
	`E / DOMAIN_NAME / DD03L-DOMNAME`
	`E / CHECK_TABLE / DD03L-CHECKTABLE`
	`E / CONT_AREA_LIST / TAB512-WA`
	`T / COLUMNS / LISTZEILE`

/BODS/TREE_NAVIGATE	Allows GUI-based navigation of an SAP application environment.
	`I / P_LANG / DD04T-DDLANGUAGE`
	`I / P_TREEAREA / DD02D-DDTEXT`
	`I / P_TREEGROUP / DD04L-ROLLNAME`
	`I / P_TREETABLE / DD02D-TABNAME`

/BODS/TREE_PROF	Reads an ERP hierarchy for profiling.
	`I / P_GROUP / DD04L-ROLLNAME`
	`I / P_TABLE / DD02D-TABNAME`
	`T / WRITES / LISTZEILE`

/BODS/TREE_SEARCH	Allows GUI-based search and selection in an SAP application environment.
	`I / TREENAME / DD08T-MESTEXT`
	`I / TREEDESC / DD04T-DDTEXT`
	`I / LANG / SY-LANGU`
	`I / MAX_ROWS / SY-TABIX`

Function name	Description

	`I / FLAG / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

/BODS/TREE_NAVIGATE40	Provides a GUI-based method to import tables and fields for SAP R/3 version 4.x and later environments.
	`I / P_LANG / DD04T-DDLANGUAGE`
	`I / P_TREEAREA / DD02D-DDTEXT`
	`I / P_TREEGROUP / DD04L-ROLLNAME`
	`I / P_TREETABLE / DD02D-TABNAME`
	`T / P_RETURN / LISTZEILE`

/BODS/TREE_IMPORT40	Provides a GUI-based method to import tables and fields for SAP R/3 version 4.x and later environments.
	`I / P_LANG / DD04T-DDLANGUAGE`
	`I / P_TREEAREA / DD02D-DDTEXT`
	`I / P_TREEGROUP / DD04L-ROLLNAME`
	`I / P_TREETABLE / DD02D-TABNAME`
	`E / DESCRIPTION / DD04T-DDTEXT`
	`E / CONTROLLING_AREA / DD02D-DDTEXT`
	`E / CHART_OF_ACCT / DD02D-DDTEXT`
	`E / TABLE_NAME / DD02D-TABNAME`
	`E / FIELD_NAME / DD03D-FIELDNAME`
	`E / DOMAIN_NAME / DD03L-DOMNAME`
	`E / CHECK_TABLE / DD03L-CHECKTABLE`
	`E / CONT_AREA_LIST / TAB512-WA`
	`T / COLUMNS / LISTZEILE`

/BODS/TREE_SEARCH40	Allows GUI-based search and selection in an SAP R/3 version 4.x environment.
	`I / TREENAME / DD08T-MESTEXT`
	`I / TREEDESC / DD04T-DDTEXT`
	`I / LANG / SY-LANGU`
	`I / MAX_ROWS / SY-TABIX`
	`I / FLAG / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

/BODS/UPLOAD	Uploads a Data Services-generated ABAP program to the SAP server.
	`I / P_PNAME / SY-REPID`
	`I / P_TITLE / RS38M-REPTI`
	`I / P_DEVC / TDEVC-DEVCLASS`
	`I / P_APPL / TRDIR-APPL`
	`I / P_RSTAT / TRDIR-RSTAT`
	`I / P_REQNUM / E070-TRKORR`
	`I / P_TSKNUM / E070-TRKORR`
	`T / PROGRAM / PROGTAB`
	`T / ENTRIES / TAB512`

/BODS/RFC_STREAM_READ_TABLE	Allows data streaming via RFC protocol.
	`I / QUERY_TABLE / TABNAME`
	`I / CALLBACK_FM / FUNCNAME`
	`I / DELIMITER / SONV-FLAG`
	`I / BATCH_SIZE / INT4`
	`I / NUMB_OF_LINES / INT4`

Function name	Description

	`T / PROJECTIONS / RFC_DB_OPT`
	`T / SELECTIONS / RFC_DB_OPT`

12.2.3 SAP applications security levels

You can choose from three security levels based on your system requirements. Typically, application development environments have four application phases: development, consolidation, test, and production. ABAP programs are generated in the development phase, consolidated from multiple developers, transported to the test phase for testing, and finally transported to the production phase. The Data Services security mechanisms were designed using the SAP application paradigm. The following table displays the various authorization levels and application phases.

User	Authorization level	Application phase
DEVUSER	High	Design or Design and Consolidation
TESTUSER	Middle	Test or Consolidation and Test
PRODUSER	Low	Production

A typical application passes through four phases:

1. In the design phase, a high-security profile user designs the data and work flows, and generates ABAP programs. The ABAP programs are dynamically communicated to SAP and executed.
   After execution, ABAP programs are automatically deleted.
   To do this, the developer must have authorization that could be designated DEVUSER.
2. In the consolidation phase, developers upload generated ABAP programs into their SAP system with predefined program names and preregistered correction numbers. A user in that phase can run these programs with a lower security profile, TESTUSER, for example.
   Consolidation can also occur in the design phase.
   Here TESTUSER can create ABAP programs, install SAP functions, and schedule jobs.
3. In the test phase, the Data Services repository must be exported to a new repository, and the generated ABAP programs must be transported to the SAP test system. This is usually performed by TESTUSER.
4. The production phase begins when no additional program modifications are required.
   The user involved in the move to the production phase has the lowest security profile. The user in this phase could be called PRODUCER.

SAP authorization can be managed in two ways:

1. Profile generation method
   Every user is assigned to activity groups. Each activity group contains information about allowed activities (such as transaction and menu path). Only users with an activity authorization can perform that activity.
2. Profiles assigned to users method
   Every profile contains authorizations. The user administrator creates new authorizations, assigns them to profiles, and assigns profiles to users.

Data Services provides various authorization choices to meet your organization's needs:

- Some organizations use DEVUSER during all the application phases.
- Others do not want to assign DEVUSER authorization, which requires a high security profile, to any individual. In Data Services you can use the TESTUSER authorization to:
  - Generate ABAP on the development system
  - Upload it to SAP, then execute
- PRODUSER typically has the lowest authorization. This user can usually only execute ABAP programs to which TESTUSER has been granted authorization.

Data Services provides the checking mechanisms necessary to prevent unauthorized users from executing Data Services-generated programs in your SAP system. If you specify a security profile in SAP datastores (see *Defining SAP Applications datastores* [page 2438]), Data Services verifies authorization before executing the program.

If you do not specify a security profile, the Data Services programs do not perform a security check prior to execution.

12.2.4 SAP user authorizations

This section describes how to define a profile using the profile generation method and the specific user authorization level method. It also lists recommended SAP applications security profiles and authorizations for using Data Services.

12.2.4.1 Creating an SAP applications profile using profile generation

1. Create a user (for example for SAP DEV, TEST, or PRD systems).
2. Open the SAP Profile Generator (transaction PFCG).
3. Create an Activity group or Role.
4. Enter a description for the role.
5. Go to the *Authorizations* tab and click *Change authorization data*.
6. On the *Change Role: Authorizations* screen, click the *Manually*, toolbar icon.
7. The *Manual Selection of Authorizations* window opens.
8. Enter the required authorizations for this role.
9. Click *OK*
10. Return to the *Change Role: Authorizations* screen.
11. Manually configure components by entering the values documented in the section *Authorizations for Data Services* [page 2423].
12. To complete the security profile, click the *Back* icon (or press F3), select the *User* tab, enter your SAP user ID for Data Services, and click the *Save* icon.

12.2.4.2 Defining a Data Services-specific authorization level

1. Determine which SAP profile you need.

   Required profiles depend on how you use Data Services. See *SAP application profiles* [page 2421].
2. In SAP, create the required authorizations for your profiles.

   The required authorizations for your profiles are listed in *SAP application profiles* [page 2421]. Settings for each authorization are listed in *Authorizations for Data Services* [page 2423].
3. Create an SAP profile.

   For example, you might create a profile specific to developing applications in Data Services: DS_DEV.
4. Assign the SAP authorizations to the SAP profile.
5. Assign the profile to an SAP user for Data Services.

12.2.4.3 SAP application profiles

How you use Data Services determines the profiles you need to define:

- Development and test profile
- Production profile

For information about the SAP BW loading profile, see *SAP NetWeaver BW loading profile* [page 2512].

12.2.4.3.1 Development and test profile

To support development and test, create an SAP profile such as DS_DEV, that defines authorizations for DEVUSER or TESTUSER. This profile requires the following authorizations.

- S_BTCH_JOB
- S_DEVELOP
- S_RFC
- S_TABU_DIS
- S_TCODE

Related Information

Authorizations for Data Services [page 2423]

12.2.4.3.2 Production profile

To support production, create an SAP profile such as DS_PROD that defines authorizations for PRODUSER. This profile requires the following authorizations:

- S_BTCH_JOB
- S_RFC
- S_TABU_DIS
- S_TCODE

Related Information

Authorizations for Data Services [page 2423]

12.2.4.4 Development versus production functions

User permissions differ between development and production environments. A user with S_DEVELOP authorization can execute any function. If that authorization isn't present, the ABAP code checks the authorization object ZSDS.

Note that you can create custom authorization checks for any function by modifying the sample ABAP code in the shipped file sample_badi_impl.txt (admin\R3_Functions\ManualInstall).

12.2.4.4.1 Development-only functions

The following functions should be used only in development environments, not on production systems:

/BODS/AUTH_IMPORT

/BODS/EXTRACTOR_IMPORT

/BODS/FUNCTION_GET

/BODS/IDOC_IMPORT

/BODS/RFC_ABAP_INSTALL_AND_RUN

/BODS/TABLE_IMPORT

/BODS/TREE_IMPORT

/BODS/TREE_IMPORT40

/BODS/UPLOAD

12.2.4.4.2 Production functions

If the system is not a development system, then the user can only execute the following limited set of functions.

/BODS/BW_QUERY

/BODS/EXTRACTOR_NAVIGATE

/BODS/EXTRACTOR_SEARCH

/BODS/FILE_ROWCOUNT

/BODS/GET_VERSION

/BODS/IDOC_SEARCH

/BODS/JOB_LOG

/BODS/JOB_STATUS

/BODS/MODEL_NAVIGATE

/BODS/READ_TEXT

/BODS/SYNTAX_CHECK

/BODS/TABLE_SEARCH

/BODS/TEXTS

/BODS/TREE_NAVIGATE

/BODS/TREE_NAVIGATE40

/BODS/TREE_SEARCH

/BODS/TREE_SEARCH40

/BODS/COLUMN_SEARCH

/BODS/ABAP_RUN

/BODS/JOB_RUN

/BODS/RFC_READ_EXTRACTOR

/BODS/RFC_READ_TABLE

/BODS/RFC_READ_TABLE2

/BODS/RFC_READ_TABLE_FILE

/BODS/TREE_PROF

/BODS/DATA_PROFILE

/BODS/RFC_STREAM_READ_TABLE

12.2.4.5 Authorizations for Data Services

This section describes the authorizations that support Data Services operations.

For improved security, avoid using wildcards, generic, or blank values for authorization fields, especially in a production environment. Enter more specific values that are appropriate to your business applications.

12.2.4.5.1 Open Hub

The necessary authorizations to use the Open Hub interface in Data Services are contained in the S_BI-WHM_RFC profile.

In addition, Data Services needs the following authorization to work with the Open Hub interface:

S_RFC_ADM

Purpose: This object includes authorization checks for accessing individual administration functions in transaction SM59

Use: DEV, PROD

Class: Cross-application Authorization Objects

Text (Description): Administration for RFC Destination

Field	Values
Activity	03
RFCTYPE	T
RFCDEST	List of RFC destinations the user is allowed to access
ICF_VALUE	Authorizations for destination in transaction SM59

12.2.4.5.2 G_800S_GSE

Purpose: This authorization allows Data Services to access ERP hierarchies.

Use: DEV, PROD

Text (Description): Special Purpose Ledger Sets: Set

Class: Financial Accounting

Field	Values
Authorization group	Not used
Activity	03

12.2.4.5.3 S_BTCH_ADM

Purpose: This authorization checks background processing privileges.

Use: DEV, PROD

Text (Description): Background Processing: background Administrator

Class: Basis

Field	Values
Background administrator ID	Y

12.2.4.5.4 S_BTCH_JOB

Purpose: This authorization checks batch job release privileges.

Use: DEV, PROD

Text (Description): Batch processing

Class: Basis

Field	Values
Job operation	RELE
Summary of jobs for a group	Not used

12.2.4.5.5 S_CTS_ADMI

Purpose: This authorization allows Data Services to perform CTS operations.

Use: DEV

Text (Description): Administration Functions in Change and Transport System

Class: Basis: Administration

Field	Values
Administration Tasks for Change and Transport System	PROJ

12.2.4.5.6 S_DEVELOP

Data Services uses the S_DEVELOP authorization in several ways.

Purpose: This implementation allows Data Services to perform a column search.

Use: DEV, PROD

Text (Description): ABAP Workbench

Class: Basis - Development Environment

Field	Values
Package	List of packages for tables that a user is allowed to access
Object type	TABL
Object name	List of tables that a user is allowed to access
Authorization group ABAP/4 program	Not used
Activity	03

Purpose: This authorization allows Data Services to run generated programs on the SAP server.

Use: DEV

Text (Description): ABAP Workbench

Class: Basis - Development Environment

Field	Values
Package	$TMP
Object type	PROG
Object name	List of temporary program names that are allowed to be generated
Authorization group ABAP/4 program	Not used
Activity	01 and 02

Purpose: This implementation allows Data Services to import a table or to search for a table.

Use: DEV, PROD (table search)

Text (Description): ABAP Workbench

Class: Basis - Development Environment

Field	Values
Package	List of packages for tables that a user is allowed to access
Object type	VIEW, TABL and TTYP
Object name	List of tables and views that a user is allowed to access
Authorization group ABAP/4 program	Not used
Activity	03

12.2.4.5.7 S_IDOCDEFT

Purpose: This authorization allows Data Services to work with IDocs.

Use: DEV, PROD

Text (Description): WFEDI: S_IDOCDEFT - Access to IDoc Development

Class: Basis - Central Functions

Field	Values
Activity	03
Extension	Not used
Basic type	Not used
Transaction Code	WE30

12.2.4.5.8 S_RFC

Purpose: This authorization allows users to execute remote functions on an SAP server.

Use: DEV, PROD

Text (Description): Authorization check for RFC access

Class: Cross-application authorization object

Field	Values
Activity	16
Name of RFC to be protected	BAPI, CADR, RFC1, SCAT, SDIF, SLST, SUNI, SUTL, SDTX, SYST, /BODS/BODS, RSAB, SDIFRUNTIME, and any other required function group
Type of RFC object to be protected	FUGR

12.2.4.5.9 S_RFC_ADM

Purpose: This authorization is required for RFC streaming

Use: DEV, PROD

Text (Description): Administration for RFC Destination

Class: Cross-application

Field	Values
Activity	03
Type of Entry in RFCDES	Not used
Logical Destination (Specified in Function Call)	RFC destination
Internet Communication Framework Values	Not used

12.2.4.5.10 S_RO_OSOA

Purpose: This authorization checks DataSource access prvileges.

Use: DEV, PROD

Text (Description): SAP DataSource Authorizations

Class: BW Service API

Field	Values
Activity	03
DataSource (OSOA/OSOD)	DataSource for data extraction
Application Component of a DataSource (OSOA/OSOD)	Not used
Subobject for DataSource	DATA

12.2.4.5.11 S_RS_ADMWB

Purpose: Use this authorization for BW loading.

Use: DEV, PROD

Text (Description): Administrator Workbench - Objects

Class: Business Warehouse

Field	Value
Administrator Workbench object	WORKBENCH, SOURCESYS, APPLCOMP, INFOAREA, INFOOBJECT, INFOPACKAG, ODSOBJECT
Activity	03

Related Information

SAP NetWeaver BW as a target [page 2512]

12.2.4.5.12 S_RS_ICUBE

Purpose: This authorization allows Data Services to access an InfoCube.

Use: DEV, PROD

Class: Business Information Warehouse

Text (Description): Data Warehousing Workbench - InfoCube

Field	Values
InfoArea	List of InfoAreas that a user is allowed to access
InfoCube	List of InfoCubes that a user is allowed to access
InfoCube Subobject	DEFINITION
Activity	03

12.2.4.5.13 S_RS_ODSO

Purpose: This authorization allows Data Services to access a DataStore Object.

Use: DEV, PROD

Text (Description): Data Warehousing Workbench - DataStore Object

Class: Business Information Warehouse

Field	Values
InfoArea	List of InfoAreas that a user is allowed to access
DataStore Object	List of DataStore Objects that a user is allowed to access
Subobject for ODS Object	DEFINITION
Activity	03

12.2.4.5.14 S_SCRP_TXT

Purpose: This authorization allows Data Services to read SAP texts.

Use: DEV, PROD

Text (Description): SAPscript: Standard text

Class: SBOP Data Services Authorization Object

Field	Values
Language Key	List of language keys that a user is allowed to access
Text ID	List of text IDs that a user is allowed to access
Name	List of text names that a user is allowed to access
Activity	SHOW

12.2.4.5.15 S_TABU_DIS

Purpose: This authorization allows Data Services to access table data in an SAP system.

Use: DEV, PROD

Text (Description): Table Maintenance (via standard tools such as SM30)

Class: Basis

Field	Value
Activity	03
Authorization group	Table groups that a user is allowed to access

12.2.4.5.16 S_TCODE

Data Services uses the S_TCODE authorization in several ways.

Purpose: This authorization grants the user access to specific transactions.

Text (Description): Authorization check for transaction start

Class: Cross-application authorization object

Field	Value
Transaction Code	SE37, SE38, SU53

Purpose: This authorization allows Data Services to execute functions in the Data Warehousing Workbench.

Use: DEV, PROD

Text (Description): Transaction Code Check at Transaction Start

Class: Cross-application Authorization Objects

Field	Values
Transaction Code	RSA1

In addition, you should have access to the contents of the following tables:

RSDAREA

RSDAREAT

RSDCUBE

RSDCUBET

RSDODSO

RSDODSOT

12.2.4.5.17 S_TRANSPRT

Purpose: This authorization allows Data Services to access the Transport Organizer.

Use: DEV

Text (Description): Transport Organizer

Class: Basis - Development Environment

Field	Values
Request Type (Change and Transport System)	DTRA
Activity	01

12.2.4.5.18 S_USER_GRP

Purpose: This authorization allows Data Services to establish a connection to the SAP server.

Use: DEV, PROD

Text (Description): User Master Maintenance: User Groups

Class: Basis: Administration

Field	Values
User group in user master maintenance	User group for Data Services user

12.2.4.5.19 S_USER_PRO

Purpose: This authorization allows Data Services to import an authorization profile.

Use: DEV

Text (Description): User Master Maintenance: Authorization Profile

Class: Basis: Administration

Field	Values
Auth. profile in user master maintenance	Authorization Profile to be imported
Activity	03

12.2.4.5.20 ZDSAUTH

Purpose: This authorization gives a user an access to Data Services functions.

Use: DEV, PROD

Text (Description): SBOP Data Services - general authorization

Class: SBOP Data Services Authorization Object

Field	Values
ACTVT: Activity	16 (Execute)

12.2.4.5.21 ZDSDEV

Purpose: This is the general authorization object (Data Services-specific version of S_DEVELOP).

Use: DEV, PROD

Text (Description): SBOP Data Services Authorization Object for development

Class: SBOP Data Services Authorization Object

Field	Values
Package	List of packages for tables that a user is allowed to access
Object type	VIEW, TABL, and TTYP
Object name	DD objects that a user is allowed to access
Authorization group ABAP/4 program	Not used
Activity	03

12.2.4.5.22 ZPGMCHK

Purpose: This authorization determines which programs may execute in a production environment.

Use: PROD

Text (Description): SBOP Data Services Authorization Object for program names

Class: SBOP Data Services Authorization Object

Field	Values
ACTVT: Activity	16 (Execute)
PROGRAM: ABAP program name	Program names that are allowed to be executed in a production environment

12.2.4.5.23 ZSDS

Purpose: This authorization lets you to define whether the SAP system should be treated as a development or production system from the perspective of Data Services.

Use: DEV, PROD

Text (Description): Data Services Authorization Object for functions

Class: SBOP Data Services Authorization Object

Field	Values
ACTVT	Not used
ZSYSTYPE	D: Development system Any other value: Production system

12.2.4.5.24 Browsing metadata for an SAP BW source datastore

In addition, if you need to be able to browse metadata for an SAP BW source datastore, you should have access to the contents of the following tables:

RSDAREA

RSDAREAT

RSDCUBE

RSDCUBET

RSDODSO

RSDODSOT

12.3 Enabling security between SAP and Data Services

12.3.1 Authentication using Secure Network Communications (SNC)

Enabling SNC provides a secure connection between SAP systems and Data Services.

This procedure describes how to configure Authentication using SNC between Data Services clients for secure communications from SAP systems.

For more information about SNC security, on the SAP Service Marketplace see the *SNC User's Guide* at *https://service.sap.com/~form/sapnet?_SHORTKEY=00200797470000074724*.

1. In Data Services, verify installation of a 64-bit SNC library.
2. Complete the procedure "Configuring the Use of the SAP Cryptographic Library for SNC" including creating a trusted certificate. This document is available on the SAP Service Marketplace Web site at *https://service.sap.com/~form/sapnet?_SHORTKEY=00200797470000074724*.
3. Add or open the datastore editor to configure and click *Advanced* to display the *Authentication* options. For *Authentication*, select *SNC*. Complete the remaining SNC options as applicable.

Related Information

Datastore [page 2531]
Secure Network Communications (SNC) in BW [page 2436]

12.3.2 Security between SAP BW and the Data Services RFC server

Data Services provides two separate options for enabling security between SAP BW and the Data Services RFC server.

- Access Control List (ACL): ACL security authorizes user access permissions to Data Services repositories through the SAP BusinessObjects Central Management Server.
- Secure Network Communications (SNC): SNC provides a secure connection between SAP BW and the Data Services RFC server.

Sample usage scenarios:

- In a private network where the SAP user's identity must be verified to access a Data Services repository, implementing the ACL option is sufficient.
- In a public network but with trusted SAP users, implemeting the SNC option is sufficient.
- In a public network where the SAP user's identity must be verified to access a Data Services repository, implement both the ACL and SNC options.

For more information about SAP security, on the SAP Service Marketplace Web site see *https://service.sap.com/security*.

12.3.2.1 Access Control List (ACL)

Enabling ACL security authorizes and authenticates users for access to Data Services repositories from SAP BW using the SAP BusinessObjects Central Management Server (CMS). The interface for the CMS is the Central Management Console (CMC). For help using the CMC, after logging in click the question mark icon.

This procedure also includes configuration steps in the SAP BW system and in the Data Services Administrator.

1. In the CMC, authenticate the SAP BW system as follows:

a) Select the *Authentication* management area from the drop-down list.

b) Double-click the *SAP* object to open it.

c) On the *Entitlement Systems* tab, select the system to authenticate and click *Update*.

d) On the *Role Import* tab, add the SAP role(s) to import (that correspond to the Data Services repositories you want to secure) and click *Update*.

e) On the *User Update* tab under *Update Roles and Aliases*, click *Update Now* to import the users associated with the imported SAP roles.

> **Note**
>
> You can also select *Schedule* to schedule regular updates to the user list.

f) Close the dialog box.

To ensure the SAP roles and users have been imported, select the CMC *Users and Groups* management area. Select the *User List* and *Group List* (or *Group Hierarchy*) nodes to view the imported users and roles. *Roles* are referred to as *groups* in the CMC.

2. In the CMC, set group and user rights for access to Data Services repositories as follows:

a) Select the *Data Services* management area from the drop-down list.

b) Open (double-click) the repository to configure.

c) Select *User Security*.

d) Click *Add Principals*.

e) Select the user or group to configure and click *Add and Assign Security*.

f) On the *Access Levels* tab, assign the desired access level(s) and click *Apply.*

g) To add or remove rights for principals, click the *Advanced* tab > *Add/Remove Rights*. You can grant only the following general global rights for Data Services:

Delete objects: Delete objects only

Edit objects: Edit, delete, or view objects

View objects: View objects only

h) To grant a principal the right to execute batch jobs in Data Services, select the *Applications* managemant area and open the *Data Services Application* object. Repeat steps 2c through 2g, except on the *Add/ Remove Rights* dialog box, you can grant the specific right *Execute batch job*.

3. In the SAP BW system, configure the profile parameters login/create_sso2_ticket=2 and login/ accept_sso2_ticket=2. You can use transaction rz11. See the *Maintain profile parameters* > *Profile parameter maintenance* interface.

For more information, refer to the procedure "Changing Profile Parameters for SSO" on the SAP Help Portal at *http://help.sap.com/saphelp_nw70/helpdata/en/45/590250736a6628e10000000a1553f6/frameset.htm*.

4. In the SAP BW system, configure RFC destination authentication as follows:

a) Use transaction sm59 to access RFC Destinations.

b) Expand *TCP/IP connections*.

c) Open the RFC destination to configure.

d) On the *Logon and Security* tab, select *Send Assertion Ticket*.

The assertion ticket contains user authentication information.

e) Save the configuration.

5. In the Data Services Administrator, enable ACL security as follows:

a) Configure the RFC server if you haven't already.

Navigate to *SAP Connections* > *RFC Server Interface* > *RFC Server Interface Configuration* > *Add*. For more information, see the Data Services *Management Console Guide*.

b) On the RFC Server Configuration page, expand the *Server security* node and select the *Enable ACL check* box.

Related Information

Secure Network Communications (SNC) in BW [page 2436]

12.3.2.2 Secure Network Communications (SNC) in BW

Enabling SNC provides a secure connection between SAP BW and the Data Services RFC server.

This procedure describes how to configure SNC between SAP BW and the Data Services RFC server for jobs that are launched from SAP BW.

1. In Data Services, verify installation of a 64-bit SNC library.
2. Complete the procedure "Configuring the Use of the SAP Cryptographic Library for SNC" including creating a trusted certificate. This document is available on the SAP Service Marketplace Web site at *https://websmp207.sap-ag.de/~form/sapnet?_SHORTKEY=00200797470000074724*.
3. Taking the results from the procedure in step 2, navigate to the Data Services Administrator *RFC Server Configuration* page to enable SNC security (*SAP Connections* > *RFC Server Interface* > *RFC Server Interface Configuration* and select the server name):
   a) Expand the *Server security* node.
   b) Select *Enable SNC*.
   c) Enter the *Server SNC name*.
   d) Enter the *SNC library path*.
   e) Select the desired *SNC quality of protection* from the drop-down list.

Related Information

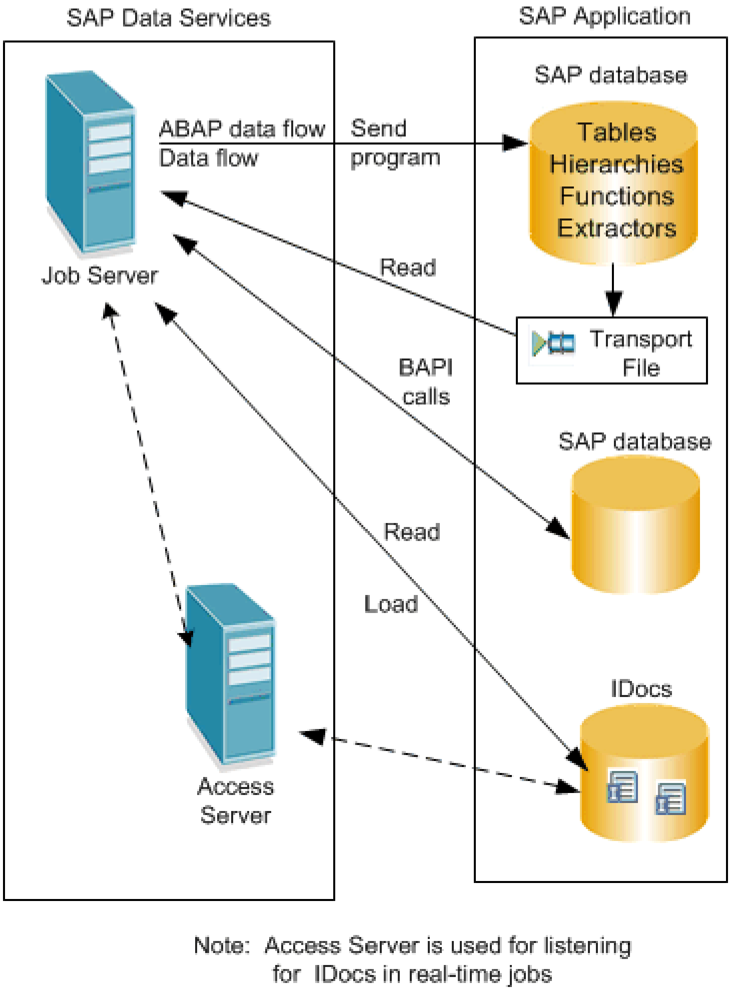
Access Control List (ACL) [page 2434]
Authentication using Secure Network Communications (SNC) [page 2433]

12.4 Connecting to SAP Applications

Data Services provides several methods for moving data into and out of SAP applications:

Application	Description
Streaming data from SAP using RFC	Reads data from SAP applications using regular data flows and supports tables and ODP sources.
ABAP programming language	Reads data from SAP applications using ABAP data flows and supports tables, hierarchies, ODP sources, and functions with scalar arguments.
RFC/BAPI	Loads data to SAP applications and allows direct access to application tables outside of the data flows through RFC function calls. Because a function is handled as a nested column within a query, it can return SAP tables, structures, or scalar parameters, which can be loaded into any target. Data Services can process any RFC-enabled function call including all available parameters plus user-supplied functions. These functions are often used to write certain information into an SAP system or read data that has to be calculated by the function.
IDoc interface	Reads from and loads data to SAP applications. Data Services can send, receive, and create SAP IDocs including extended and custom IDocs and supports reduced input sets, parallel processing, real-time, and batch processing.

The following diagram shows a high-level overview of the components involved with reading data from and loading data to SAP applications.



You can search an SAP application server for tables, IDocs, functions, hierarchies, and ODP sources. You import metadata using an SAP Applications datastore connection.

Once you have imported metadata, Data Services provides:

Objects	Description
ABAP data flows	Move table and hierarchy data between SAP applications and Data Services using ABAP.
Data flows in batch jobs	Can contain ABAP data flows (as sources), tables, hierarchies, and ODP objects (as sources), or IDocs (as sources or targets), and SAP RFC-enabled functions.
Data flows in real-time jobs	Can contain SAP tables, IDocs as sources and targets, and RFC-enabled functions.

12.4.1 SAP Applications datastores

This section discusses the steps required to use an SAP application as a Data Services data source:

- Defining SAP Applications datastores
- Browsing, searching, and importing metadata from SAP applications

12.4.1.1 Defining SAP Applications datastores

An SAP application datastore includes the following information:

- Connection information including the application server name, the language used by the SAP client application, the client and system numbers, and optionally reference to an sapnwrfc.ini file.
  You supply these values based on your installation of the SAP application.
- Data transfer method used to exchange information between Data Services and the SAP application.
  Select the method and supply the appropriate information based on decisions you make about running Data Services jobs.
- Security information, specifically the SAP security profile to be used by all connections instigated from this datastore between Data Services and the SAP application. You supply a security profile based on your installation of the SAP application.

Related Information

Data transfer methods [page 2479]
SAP applications security levels [page 2419]

12.4.1.1.1 To define an SAP Applications datastore

1. In the object library, select the *Datastores* tab.
2. Right-click inside the object library window and choose *New*.

   The datastore editor opens.
3. Enter a unique name for the datastore in the *Datastore name* box.

   The name can contain alphanumeric characters and underscores. It cannot contain spaces.
4. For *Datastore type*, select *SAP Applications*.
5. Enter the *Application server* name.
   Refer to your SAP documentation for the appropriate syntax.
6. Enter the *User name* and *Password* information.
7. You can either save the datastore or add more information to it:
   - To save the datastore and close the Datastore Editor, click *OK*.
   - To add more parameters, click *Advanced*.
     To enter values for each configuration option, click the cell next to each configuration name. For details, see the *SAP Data Services Reference Guide*.

Related Information

Datastore [page 2531]

12.4.1.1.2 Using the sapnwrfc.ini file

To change or add parameters without modifying the datastore settings, you can specify them in an `sapnwrfc.ini` file.

Place your sapnwrfc.ini file in the same directory as the RFC client/server program (for Data Services, commonly `%LINK_DIR\bin`). Or you can specify the DS_NWRFC_INI environment variable in the operating system. For example, you could can place the sapnwrfc.ini file in `C:\Temp` and specify `DS_NWRFC_INI=C:\Temp`.

You can specify an `sapnwrfc.ini` file for two different applications in Data Services:

- To use an sapnwrfc.ini file to override datastore settings, edit the SAP datastore in the Designer. From the object library, open the editor for the SAP datastore to configure and click *Advanced*. Under the *SAP* option category, for *Use sapnwrfc.ini*, select *Yes* and enter the appropriate *Destination name*.
- To use an sapnwrfc.ini file for real-time IDocs, configure the RFC client in the Administrator. Expand the *Real-Time* node, and for the Access Server to configure, click *Client Interfaces*. Open an instance of the RFC client, and select the check box to *Use sapnwrfc.ini* and enter a destination name.

> i Note
>
> Data Services ignores the parameters USER, PASSWD, and CLIENT in sapnwrfc.ini because storing these security-related parameters could introduce a security risk.

For more information about how to use the `sapnwrfc.ini` file, refer to the SAP documentation at *http://help.sap.com/saphelp_nwpi711/helpdata/en/48/ce50e418d3424be10000000a421937/frameset.htm*.

12.4.1.1.3 SAP security profiles

You can associate a security profile with a datastore so that data flows that access SAP application sources defined by the datastore include appropriate authorization checking.

SAP security profiles are useful for:

- Creating profiles of users that can access particular ABAP programs
- Moving ABAP programs from development to test and production

Specify a security profile in the Advanced options in the datastore editor. By default, Data Services does not use any SAP security profile.

Specify any security profile defined in SAP (a predefined profile or a profile you defined). Authorization checks are embedded in generated ABAP code.

Related Information

SAP user authorizations [page 2420]

12.4.1.2 Browsing, searching, and importing metadata from SAP applications

The processes for browsing, searching, and importing metadata from SAP applications are similar to the processes you would use for any source system.

There are several types of SAP objects for which you can import metadata into the Data Services object library. You can import metadata for each one by name. You can import metadata for hierarchies, IDocs, and tables by searching. You can also browse SAP applications to find hierarchies and tables.

SAP item	Methods for importing metadata
Functions	Name
Hierarchies	Name, search, browse
IDocs	Name, search
Tables	Name, search, browse
ODP sources	Name, search, browse

12.4.1.2.1 Importing metadata by name

1. In the object library, select the *Datastores* tab.
2. Right-click the SAP Applications datastore you want to use, and select *Import by Name*.
   The *Import by Name* window opens.
3. In the *Type* box, select the SAP item to import.
4. In the *Name* box, enter the name SAP uses for the item you want to import. For example, enter **`SISC001`** for an IDoc.

   > i Note
   >
   > Hierarchies require entering a controlling area and set table.

5. Click *OK*.

12.4.1.2.2 Importing table and hierarchy metadata by browsing

Tables and hierarchies can be imported by browsing the data available through the SAP datastore.

For more information about importing SAP application hierarchies, see *Extracting data from SAP application hierarchies* [page 2456].

To import table and hierarchy metadata by browsing:

1. In the object library, click the *Datastores* tab.
2. Right-click the datastore you want to use, and choose *Open*.
   The items already imported from the datastore appear in the workspace.
3. Click *External Data*.

   Click the plus sign next to a folder icon to navigate the structure of tables or hierarchies. Tables appear with a table icon.

   Hierarchies are also nested according to their use in SAP.
4. Select the items for which you want to import metadata.

   For example, to import a table, you must select a table rather than a folder that contains tables. To import a hierarchy, select the hierarchy group that contains the hierarchy.
5. Right-click and choose *Import*.
6. In the object library, expand the datastore to display the list of imported objects.

12.4.1.2.3 Importing IDoc metadata using the Search option

To use IDocs, you must first import the metadata from SAP into your repository. IDocs can be imported by name or by using the Search option.

1. In the object library, click the *Datastores* tab.
2. Right-click an SAP Applications datastore and select *Search*.
   The *Search* window opens.
3. Verify the datastore or repository in which to search displays in the *Look in* box, select *External data*, and for *Object type* select *IDocs*.
4. Enter search text in either the *Name* or *Description* box:
   - In the *Name* box enter the name or part of the name of the IDoc you want to find.
   - In the *Description* box, enter a full or partial description instead of the name.
5. Click *Search*.
   A list of objects appears in the window.
6. Right-click the name of the IDoc you want to import, and choose *Import*.

   The IDoc appears in the object library under its datastore and under the IDoc category.

   Data Services imports the schema for the IDoc type, maintaining hierarchical relationships among the data fields. Double-click the IDoc type in the object library to display the IDoc schema.

12.4.1.2.3.1 Adding an IDoc to a job

1. Select an imported IDoc from the object library.
2. Drag the IDoc into a data flow.
3. Select an appropriate option from the pop-up menu: *Make IDoc file source*, *Make IDoc file target*, *Mark IDoc message source*, or *Make IDoc message target*.

12.4.2 Reading from SAP ODP sources

This section describes how to browse, import, and read from SAP ODP sources.

Data Services can read from ODP sources in the following ways:

Method	Description
ODP data sources	Refers to ODP sources that use the Operational Data Provider data replication API (ODP data replication API). In Data Services, you can browse the hierarchy of applications. Data Services supports all types supported by the ODP data replication API in addition to changed-data capture (CDC) functionality (referred to as “delta” in SAP applications).
Native	Refers to extractors that do not use the ODP data replication API. Data Services supports the TRAN and ATTR types only, and changed-data capture is not available.

Related Information

ODP source [page 2557]

12.4.2.1 Browsing ODP sources

Use an SAP Applications datastore to connect to the ODP source. Then in the object library on the *Datastores* tab, right-click the datastore and click *Open* (or double-click the datastore) to view the datastore explorer.

With *External Metadata* selected, expand the ODP node to display the available ODP sources organized in following subfolders:

Folder	Description
All	Displays an alphabetical list of all ODP sources
By Application	Displays the ODP sources hierarchically by application and type

Related Information

To define an SAP Applications datastore [page 2439]

12.4.2.2 Importing ODP source metadata

You can import ODP source by name, by browsing, or by searching.

To import by browsing, in the datastore explorer with *External Metadata* selected, right-click an ODP source and click *Import*.

In the dialog box, enter a unique combination for *Name of consumer* and *Name of project* to identify the subscription for this ODP source.

- *Name of consumer*: Use a system name such as “Data Services”.
- *Name of project*: For a given consumer (such as “Data Services”), you have the option of creating several independent subscriptions. For example, you might want to access the same DataSource in full or changed-data capture (delta) mode with different selection criteria. Therefore, enter a unique project name here to identify this subscription.

For example, for the *Name of consumer* “Data Services” and *Name of project* “Extraction for Data Warehouse 1”, you might create initial (full) and CDC data flows for a customer dimension table. Both data flows would use this same instance (subscription) of the ODP source. But for a second project that loads another data warehouse, you could import the ODP source a second time using the same consumer name but a different project name such as “Extraction for Data Warehouse 2” to make the two subscriptions independent.

For the *Extraction mode* options, if the ODP source is not delta-enabled in the SAP system, the only option is to import the ODP source in *Query* mode. These ODP sources do not support changed-data capture. In Query mode, the ODP source returns all data and pushes down WHERE clauses.

If the ODP source is delta-enabled, *Changed-data capture (CDC)* is the default extraction mode. Here you can optionally configure changed-data capture (delta) parameters as described in the next section.

To import multiple ODP sources, Ctrl-click the ODP source, right-click, and click *Import*. You can choose between *Changed-data capture* or *Query* mode for this set of ODP sources. In this case, configuration details for changed-data capture (CDC) are not available, but you can configure them for each ODP source after importing.

After importing the ODP source, you can also view properties such as the ODP source type by right-clicking the object in the object library and clicking *Properties*. To view column properties, open the ODP source, right-click a column name, and select *Properties*.

Related Information

Browsing, searching, and importing metadata from SAP applications [page 2440]

12.4.2.3 Capturing changed data

Data Services supports changed-data capture (CDC) for ODP sources enabled with the Operational Data Provider (ODP) data replication API. You can configure different CDC options in several places:

- When importing or reimporting the ODP source metadata from the datastore explorer
- On the ODP source object properties *Extraction process* tab
- After importing the object into a data flow, in the object's source properties

12.4.2.3.1 Importing for CDC

When you import an ODP data source, the *Importing (ODP source name) ODP source* dialog box displays. Define the subscription by entering a *Name of consumer* and a *Name of project*.

To optionally configure CDC for the ODP source during import:

1. Verify that the Extraction mode *Changed-data capture (CDC)* is selected.
   The table for defining CDC filters is enabled.
2. For each filter, select a *Field name* using the drop-down box in the first cell.
3. The comparison operators include:
   - Equals
   - Contains pattern
   - Between
4. Enter value(s) in the *From* and (optionally) *To* fields.
   Values are constant literals and cannot be variables or expressions.
5. Repeat the *Field name*, *Sign*, *Comparison*, and *From/To* selections for each filter.
6. Click *Import*.

The object imports with a name of the format `<ExtractorName_NameofProject>`.

When reimporting an ODP source (right-click the object name and select *Reimport*), you can also change the CDC filters at that time (however at that time the name fields cannot be changed).

Related Information

Importing ODP source metadata [page 2443]

12.4.2.3.2 Editing CDC filters

After importing the ODP source, you can also edit the CDC filters in the object properties.

1. In the object library, right-click the ODP source name and select *Properties*.
2. Select the *Consumer details* tab.
   The *Name of consumer* and *Name of project* fields cannot be changed.
3. Edit the *Field name*, *Sign*, *Comparison*, *From*, and *To* values for each filter.
4. Click *OK*.

If you change the filter properties for an ODP source, “reset” the job by running it with the source option *Initial load* selected. Then you can run the job with CDC filters applied (with *Initial load* cleared).

12.4.2.3.3 CDC columns

The import operation adds the following columns to the original table:

Column name	Description	Source
DI_SEQUENCE_NUMBER	Contains an integer for sequencing.	Data Services
DI_OPERATION_TYPE	Contains the row operation type: I (insert), D (delete), X (unknown)	Data Services
ODQ_CHANGEMODE	For CDC, identifies the row type. Works in conjunction with ODQ_ENTITYCNTR to determine how to handle the row. C (create), U (update), D (delete)	SAP
ODQ_ENTITYCNTR	For CDC, an integer that works in conjunction with ODQ_CHANGEMODE to determine how to the handle row.	SAP

The following table describes the relationships between ODQ_CHANGEMODE and ODQ_ENTITYCNTR.

CHANGEMODE	ENTITYCNTR	Description
C	+1	The record is new and inserted.
U	+1	The record is inserted.

CHANGE-MODE	ENTI-TYCNTR	Description
		If there is a record with same key already available, the record is updated. In this case, there should be a corresponding U- (short notation for U and -1) record. The record contains the new information.
U	-1	The record is deleted and replaced with the corresponding record with same key and U + (short notation for U and +1) if available. The record contains the previous (old/ outdated) information.
U	0	The record is updated, but the key figures must be added to the already available key figures. So the record returns the difference between the old record and the new record, not the new record itself.
D	-1	The record is deleted. It contains the same key as an already existing record, but the key figures are inverted. So the record might be "deleted" by adding the key figures to the already available values. They should sum up to 0.
D	0	The record is deleted. It contains the actual data in both key and key figures.

12.4.3 Reading from SAP tables

You can use a regular data flow to process large volumes of data from SAP tables.

To improve the performance while reading large volumes of data, the source table editor includes the following options:

- The *Enable partitioning* option allows the software to read R/3 table data using the number of partitions in the table as the maximum number of parallel instances. The *Partition type* option in the *Partition* tab on the *Properties* window for a table must be set to Range or List in order to use the *Enable partitioning* option.
- The *Array fetch size* option allows the data to be sent in chunks, which avoids large caches on the source side.
- The *Execute in background (batch)* option lets you run the SAP table reader in batch mode (using a background work process) for time-consuming transactions.

Related Information

Source [page 2541]
Batch execution mode [page 2488]
Open Hub Table source [page 2565]

12.4.4 ABAP data flow processing

In Data Services, ABAP data flows extract SAP data from tables and follow this general execution process:

- In a development environment, Data Services generates the ABAP code, or in production, the program is uploaded to the SAP server
- Data Services connects to the SAP application server through remote function call (RFC).
- The SAP application server executes the ABAP code.
- The ABAP program results are generated and communicated back to Data Services, which then loads the target data cache.

> **Note**
>
> Some operations such as uploading generated ABAP programs or using the Direct Download data transfer method require interoperability with the SAP GUI. If you experience connection problems with these operations, implement the appropriate Support Package as described in *SAP Note 1258742*, “Starting SAPGUI using external RFC libraries”.

12.4.4.1 ABAP interface objects and options

The following Designer objects and options pertain to SAP.

12.4.4.1.1 Data flow objects

The Data Flows tab of the object library includes both the *Data Flows* and *ABAP Data Flows*.

You can place ABAP data flows into batch jobs as a source. ABAP data flows generate ABAP code. In most cases, you use a data transport object as the target of an ABAP data flow. A data transport specifies a staging file for data extracted from SAP applications.

ABAP data flows and data transport objects also appear in the tool palette.

Related Information

SAP-specific objects [page 2552]

12.4.4.1.2 Data flow commands

With a data flow in the workspace, the following commands appear in the *Validation* menu:

- *Validate* > *Validate ABAP*
  Validates the ABAP code Data Services produces for the data flow
- *Generate ABAP*
  Displays the ABAP generated for the current data flow

12.4.4.1.3 SAP environment options

In the Designer, the following options are available under *Tools* > *Options* > *SAP* > *Environment*.

Option	Description
Maximum number of rows buffered before transfer to Data Services	The maximum number of rows held in the SAP buffer prior to writing to the data transport file. The default value is 5000 rows. This option lets you manage the amount of memory used by generated ABAP programs by selecting the number of rows to write in a given batch. You set this value once and it applies to all SAP application connections initiated by Data Services.
Prefix for ABAP program names	You can define this parameter to separate Data Services-generated ABAP program names from other ABAP program names. Because ABAP program names must begin with a Y or a Z, this option provides a default that begins with ZW. The maximum for this option is two characters.
Convert SAP null to null	Converts NULL values from SAP sources in data flows to database-specific NULL.
Disable background SAP job status in Data Services log	Disables trace log entries when Data Services queries the status of jobs submitted in background (using the datastore option *Execute in background (batch)*).

Related Information

Working with null values from SAP applications [page 2581]

12.4.4.1.4 Data transfer requirements

When you configure SAP Applications datastores in Data Services, you specify the data transfer method. This method defines how data that is extracted by the ABAP running on the SAP application server becomes available to the Data Services server (the computer on which the Data Services Job Server runs). The options are:

Option	Description
RFC	Use to stream data from the source SAP system directly to the Data Services data flow process using RFC.
Direct download	The SAP server transfers the data directly to the *Local directory* using the SAP-provided function GUI_DOWNLOAD or WS_DOWNLOAD.
Shared directory	Default method. The SAP server loads the transport file into the *Working directory on SAP server*. The file is read using the *Application path to the shared directory* from the Job Server computer.

Option	Description
FTP	The SAP server loads the *Working directory on SAP server* with the transport file. Then the Job Server calls an FTP program and connects to the SAP server to download the file to the *Local directory*.
Custom Transfer	SAP server loads the *Working directory on SAP server* with the transport file. The file is read by a third-party file transfer (custom transfer) program and loaded to the *Custom transfer local directory*.

Related Information

Data transfer methods [page 2479]

12.4.4.1.5 SAP applications permissions

Data Services uses different execution methods to execute generated ABAP in design, development, test, and production environments. Each method has different security requirements.

Related Information

SAP applications security levels [page 2419]

12.4.4.1.6 Security profile

ABAP programs generated by Data Services can verify the permissions of the user executing the program. If a security profile is provided, the generated ABAP verifies that the user executing the program has the permissions specified by that profile. This security profile should be transported to the production system prior to running Data Services against it.

12.4.4.2 Data flows

SAP applications commonly consist of three key components:

Component	Description
SAPGUI client	A front-end application that provides a logical view of the entire SAP system

Component	Description
Application layer	An application server that interprets ABAP, SAP's proprietary fourth-generation language, and generates and executes SQL statements against the database
Database layer	A database manager that manages the SAP data in its tables

Interacting directly with the SAP application database layer can be problematic because:

- It is complex.
- The data contained in pool and cluster tables can be accessed only through ABAP, not through SQL.

The Data Services SAP ABAP interface allows you to extract SAP application data, through the application layer, using ABAP. This interface also provides the ABAP data flow and its associated Data Services Designer options. This means that:

- You do not need to understand many of the complexities of the SAP database layer.
- You can use Data Services to access not only transparent tables, but also pool and cluster tables.
- You can call corresponding SAP functions to ensure the results match what SAP would display.

To use an ABAP data flow, create a batch job, create a data flow inside the job, then create an ABAP data flow inside the first data flow. This type of batch job uses the following general execution process:

- Data Services generates the ABAP code.
- Data Services connects to the SAP application server via remote function call (RFC).
- The ABAP code executes on the SAP application server.
- The ABAP program results are generated and communicated back to Data Services, which then loads the target data cache.

Related Information

Designer Guide: Naming data flows [page 281]

12.4.4.2.1 What is an ABAP data flow?

An ABAP data flow extracts and transforms data from SAP application tables, files, and hierarchies. The ABAP data flow produces a data set that you can use as input to other transforms, save to a file that resides on an SAP application server, or save to an SAP table. When Data Services executes ABAP data flows, it translates the extraction requirements into ABAP programs and passes them to SAP to execute.

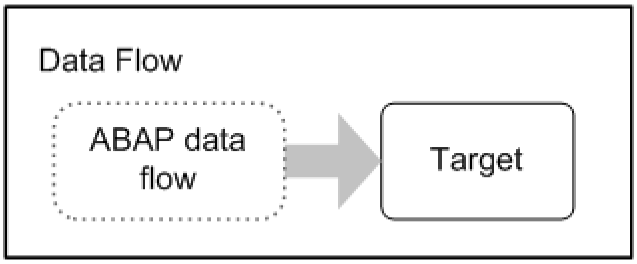
When you plan to use the SAP extracted data outside the ABAP data flow, the ABAP data flow can place the extracted data set in a file. A Data Services data transport passes the data in the file to the parent data flow. A data transport indicates how data is communicated between the SAP application server and Data Services.

When the target inside the ABAP data flow is the final destination for the data, you do not need a data transport. The ABAP data flow stores the data in a file saved on the SAP application server. In this case, the ABAP data flow must be the only object in a data flow.

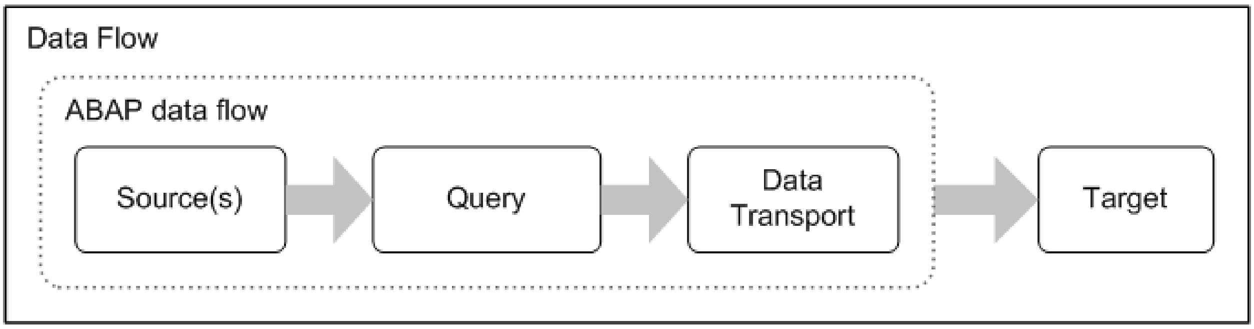
You can also use a data flow to extract data from an SAP table. When you use a data flow, Data Services extracts data from SAP using an RFC (remote function call). However, when you use an ABAP data flow, Data Services extracts data using ABAP and performance is generally better.

Because SAP performs all operations in an ABAP data flow, you want the operations in an ABAP data flow to make best use of SAP system capabilities. Therefore, the preferred method of use involves ABAP.

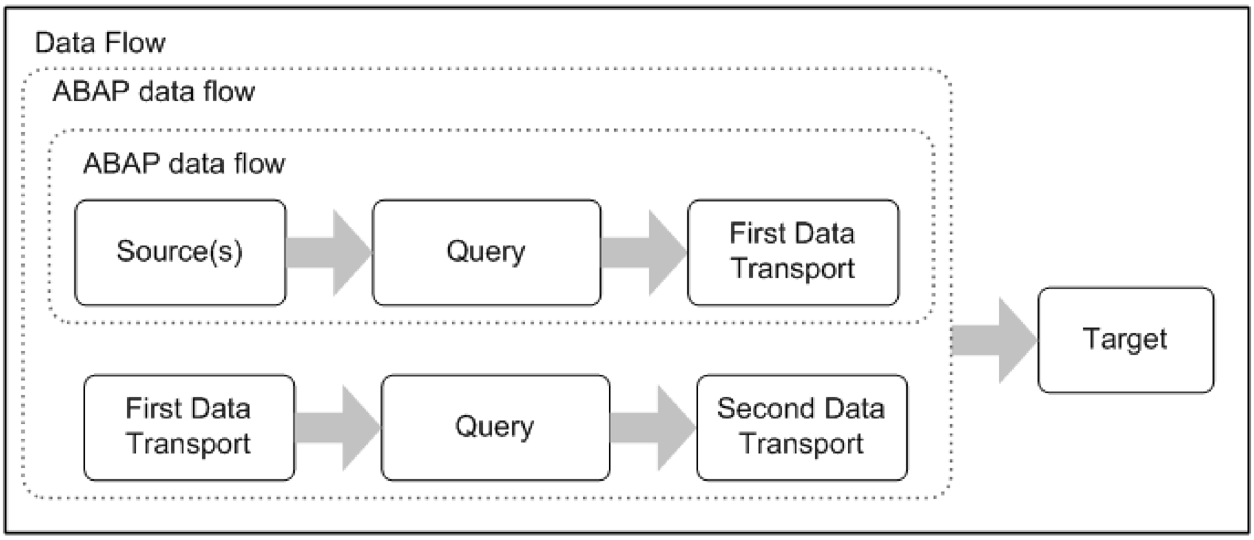
Consider the simple case of extracting data from an SAP application table and loading it into a non-SAP target. The parent data flow contains two objects: an ABAP data flow and a target.



The ABAP data flow defines the specific source tables from which you want to extract data and contains a data transport for the resulting data set.



In some cases, the transformations you need to perform in the SAP application are too complex to include in just one ABAP data flow. For example, the transformation might require multiple passes over the data. Data Services allows you to nest an ABAP data flow inside another ABAP data flow so that you can simplify each data flow. You can keep the output from the inner data flow in a file on the ABAP server and use the file in the outer data flow.



Note

Although ABAP data flows are used in data flows, data flows cannot be placed in ABAP data flows. As shown above, only other ABAP data flows can be placed inside ABAP data flows. Also, ABAP data flows cannot be used as embedded data flows.

12.4.4.2.2 Defining ABAP data flows

This section describes the four procedures to complete when defining an ABAP data flow:

- Creating R/3 data flows.
- Specifying sources
- Defining a query
- Specifying a target

12.4.4.2.2.1 Adding an ABAP data flow

This procedure describes how to first add and configure the data flow that will contain the ABAP data flow.

1. In the workspace, add a new data flow and open it.
2. On the tool palette, click the icon for an ABAP data flow.
3. Click a location in the workspace.
   The *Properties* window appears, prompting for the ABAP data flow options.
4. In the *Datastore* list, select the SAP datastore to use for the extraction.
5. In the *Generated ABAP file name* box, enter the file name for the ABAP program that this data flow generates. Data Services writes this file in the generated ABAP directory specified for the datastore.
6. Enter other options in the window.

- *ABAP program name*
  Specifies the name Data Services uses for the ABAP program name that runs in the SAP application. It must begin with Y or Z.
- *Job name*
  Specifies the name used for the job that runs in the SAP application. It defaults to the name of the data flow.
- *ABAP row limit*
  Can be used for the test phase of development to reduce the number of data rows sent to the transport file. The default value of zero indicates that all data rows will be output to the transport file. When this value is set to a number greater than zero, only the specified number of rows go to the transport file.

  If you enter a number here, you can then enable/disable this feature from the *Tools* > *Options* control panel. The Designer and Job Server must be on the same computer. ABAP row limit is enabled/disabled globally for all SAP jobs associated with this Job Server. Remember to disable this feature before moving the jobs into a production environment.

  To enable or disable the option, select *Tools* > *Options* > *Job Server* > *General*. Enter data (case sensitive) as follows:

Field	Entry
Section:	AL_Engine
Key:	ABAP_ROW_LIMIT
Value:	TRUE

  Then click *OK*.
- *Join rank*
  If the calling data flow uses output from this ABAP data flow (in the form of a data transport) in a join, join rank indicates the rank of the source relative to other sources in the data flow. The software joins sources with higher join ranks before joining sources with lower join ranks.
  Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify a join rank only in the Query transform editor.
  Join rank must be a non-negative integer. Default value is 0.
- *Parallel process threads*
  Specifies the number of threads that read or load data. This option allows you to parallel process reading and loading.
- *RFC data transfer package size*
  With the *Data transfer method* of *RFC* selected at the datastore level, you can adjust the number of rows to process per data package to maximize streaming performance. Defaults to 5000 rows.
- *Cache*
  If the calling data flow uses output of the ABAP data flow as the inner source for a join, you can cache the output to improve the performance of the join. Use caching only if you are sure the data is small enough to fit in the memory allotted to the cache.
  Default setting is No.
  The option is ignored if the ABAP data flow is nested inside another ABAP data flow.

7. Click *OK*.

   An ABAP data flow appears in the workspace and the project area.
8. With the ABAP data flow name highlighted, type a new name.
9. Click the ABAP data flow name.

   Data Services opens the workspace where you next add and define the components of the ABAP data flow.

12.4.4.2.2.2 Specifying sources

You can add SAP application tables or hierarchies to an ABAP data flow as sources. This example adds a table.

Do not use non-SAP or IDoc sources here; add those sources to the parent data flow and use a query (including BAPI functions as needed) to join those sources with the data set produced by the ABAP data flow.

To specify data sources:

1. From an SAP Applications datastore, click an SAP table and drag it into the workspace of an open ABAP data flow.
2. Click the name of the source table in the workspace to open the source table editor.
   In the source table editor, you can set three options:
   - *Join rank*
     Indicates the rank of the source relative to other tables joined in the data flow. The software joins sources with higher join ranks before joining sources with lower join ranks.
     Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor.
     Must be a non-negative integer. Default value is 0.
   - *Cache*
     Indicates whether Data Services should read the required data from the table and load it into memory. Best practice is to specify the cache in the Query transform editor. When sources are joined using the Query transform, the cache setting in the Query transform takes precedence over the setting in the source. In the Query editor, cache is set to Automatic by default. The automatic setting carries forward the setting from the source table.
     There are three options in sources:

Cache option	Description
Yes	The table is always cached unless it is the outer-most table in a join.
No	The table is never cached. Package size operation is bypassed.
Automatic	Data Services determines whether or not to cache the table based on table size and number of filtered records.

     The default option is No.
   - *Package size*
     The package size option limits the number of rows brought into memory when building internal tables. By limiting the number of rows, less memory is used to construct internal tables.
     You should use this option only for cached tables whose size could lead to excessive memory consumption.

Related Information

Extracting hierarchy set data [page 2457]
Optimizing inner joins [page 2467]
Manually setting join order [page 2471]

Tables used in a join [page 2473]

Source [page 2541]

12.4.4.2.2.3 Defining a query

Define a query to produce the desired data set. With a query, you can manipulate SAP application sources as follows:

- Join the data sets from more than one source
- Select the rows to extract from the source
- Define the schema of the extracted data set
- Perform calculations on column values

A single query can have several inner joins or several left outer joins, but may not contain mixed inner joins and left outer joins.

Best practice when joining multiple sources is to set join rank and cache for all sources in the Query editor.

When sources are joined using the Query transform, the cache setting in the Query transform takes precedence over the setting in the source. In the Query editor, cache is set to Automatic by default. The automatic setting carries forward the setting from the source table. The following table shows the relationship between cache settings in the source, Query editor, and whether the software will load the data in the source table into cache.

Cache Setting in Source	Cache Setting in Query Editor	Effective Cache Setting in ABAP data flow
Yes	Automatic	Yes
No	Automatic	No
Automatic (Data Services determines whether or not to cache the table based on table size and number of filtered records)	Automatic	Automatic (based on table size and number of filtered records)
Yes	Yes	Yes
No	Yes	Yes
Automatic	Yes	Yes
Yes	No	No
No	No	No
Automatic	No	No

Note

If any one input schema has a cache specified in the Query editor, the software considers only Query editor cache settings and ignores all source editor cache settings.

Data Services supports using the results of an ABAP data flow inside another ABAP data flow. If you are using the data in another ABAP data flow, make sure to:

- Keep the data on the SAP application server rather than on the Data Services server.
- Use *FTP* or *Shared directory* as the data transfer method.

You specify the location of the output file and data transfer method in the SAP datastore associated with the ABAP data flow.

To define a query:

1. Add a query to the ABAP data flow.
2. Draw the data flow connections.
3. Use the query editor to specify the desired data set.

Related Information

Reference Guide: Transforms, Platform Transforms, Query [page 1441]

12.4.4.2.2.4 Specifying a target

Specify a target for the resulting data set. Potential targets include a data transport, which makes the data set available to the calling data flow, or a file on the SAP application server.

To define a data transport:

1. Add a data transport to the ABAP data flow and rename it.
2. Connect the components of the ABAP data flow and map the query.
3. Click the name of the data transport.
   The data transport editor appears in the workspace.
4. In the data transport editor, enter the *File name* of the transport file.
   The file is placed in the *Working directory on SAP server* specified in the datastore definition.
5. If necessary, you can specify that the output file will be appended to an existing file with the same name (presumably created in a previous run of this same job). The default behavior is to replace any existing file.
6. Click the back arrow to return to the ABAP data flow.

   Upon execution, the ABAP data flow populates the file designated in the *File name* box.

12.4.4.2.3 Extracting data from SAP application hierarchies

The Data Services hierarchy object can extract set hierarchy information from the following set types:

Set type	Description
Basic sets	Sets containing values from one dimension. For example, in a basic set such as Assets that uses the dimension Accounts, all values in Assets are account numbers or ranges of account numbers.

Set type	Description
Single-dimension sets	Combinations of basic sets and/or other single-dimension sets that contain values from the same dimension. For example, the basic set Assets can be combined with the basic set Liabilities because they both use the same dimension Accounts.

In particular, Cost Center and Cost Element hierarchies in the Controlling module of SAP ERP are supported set hierarchies.

The following sections describe how hierarchies are defined in SAP applications and how Data Services translates the hierarchical relationships into horizontal or vertical representations:

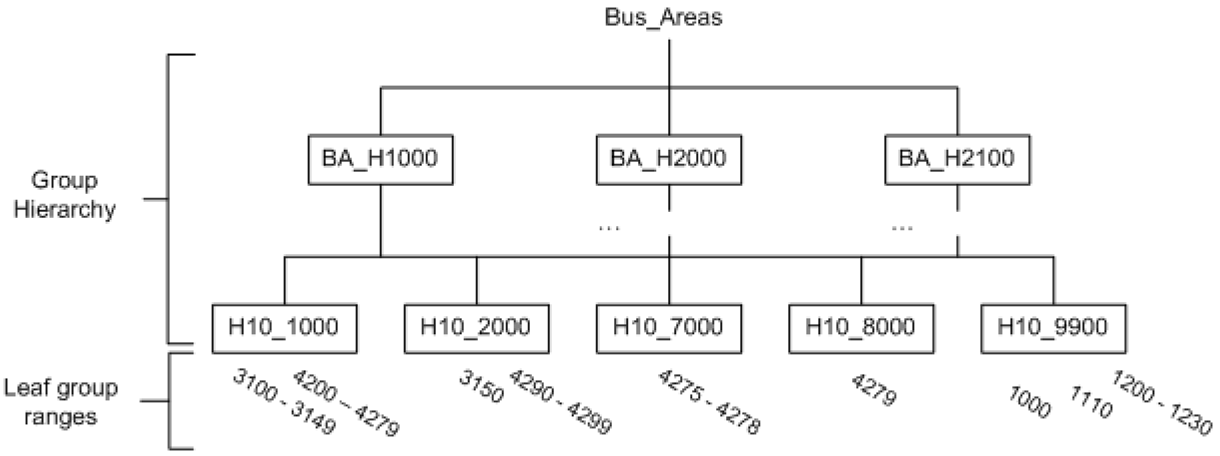
Related Information

SAP application hierarchy sets [page 2457]
Extracting hierarchy set data [page 2457]
Example data flow with an SAP source [page 2459]

12.4.4.2.3.1 SAP application hierarchy sets

Hierarchy set information from SAP applications includes hierarchy nodes and ranges of values associated with the leaf nodes. The possible values are defined by the domain upon which the hierarchy is based. The values themselves are stored in the set table associated with the hierarchy.

The set data represented in this hierarchy is as follows:



12.4.4.2.3.2 Extracting hierarchy set data

Data Services provides a hierarchy object that extracts group nodes and leaf group ranges from supported SAP set hierarchies. When Data Services extracts the hierarchy using a hierarchy object, the result includes a row for each parent-child relationship in the hierarchy and associated leaf group ranges.

A hierarchy object produces the following output when extracting the Business area hierarchy from Controlling area 1000 shown previously.

	Parent_ID	Child_ID	VALUE_FROM	VALUE_TO
NULL parent to root node		0HBUS_AREAS		
Root node to first child	0HBUS_AREAS	0HBA_H1000		
First child (now parent) to its first child	0HBA_H1000	0HH10_1000	0000003100	0000003149
	0HBA_H1000	0HH10_1000	0000004200	0000004279
First child (now parent to its second child	0HBA_H1000	0HH10_2000	0000003150	0000003150
	0HBA_H1000	0HH10_2000	0000004290	0000004299
First child (now parent) to its third child	0HBA_H1000	0HH10_7000	0000004275	0000004278
First child (now parent) to its fourth child	0HBA_H1000	0HH10_8000	0000004279	0000004279
First child (now parent) to its fifth child	0HBA_H1000	0HH10_9900	0000001000	0000001000
	0HBA_H1000	0HH10_9900	0000001110	0000001110
	0HBA_H1000	0HH10_9900	0000001200	0000001230
Root node to second child	0HBUS_AREAS	0HBA_H2000		
Second child (now parent) to its first child	0HBA_H2000	0HH20_3000	2-4200	2-4200
	0HBA_H2000	0HH20_3000	2-4210	2-4210
Root node to third child	0HBUS_AREAS	0HBA_H2100		
Third child (now parent) to its first child	0HBA_H2100	0HH21_9000	21-3100	21-3100
			...	

When you view the SAP hierarchy types in Data Services, you do not see the data associated with the hierarchy; instead, you see all of the hierarchy types grouped in two ways:

- Together in a single section (all hierarchy types organized across controlling areas by extraction logic)
- By controlling area

The names of the hierarchy types are made up of the SAP application set table name and the hierarchy group name. The hierarchies are named after the root node.

To extract data using a hierarchy, import the hierarchy metadata as part of an SAP application datastore definition.

To import a hierarchy type from an SAP application to your repository:

1. In the object library, select the *Datastore* tab and open the datastore corresponding to the SAP instance from which you want to import the hierarchy type.
2. In the workspace select *External Data*.
3. Expand the Hierarchy node.

4. Select the hierarchy type to import.

   The available hierarchies are listed according to the controlling areas with which they are associated. You can also browse through all hierarchies regardless of controlling area in the section labeled “All Controlling Areas”.
5. Right-click and choose *Import*.

The internal view of the datastore shows the hierarchy type.

12.4.4.2.3.2.1 The imported hierarchy

After a hierarchy type is imported into a datastore, it appears in the object library.

12.4.4.2.3.2.2 A hierarchy instance in an ABAP data flow

Use a hierarchy object as you would a table from the object library: select the hierarchy, then drag it into the definition of the ABAP data flow open in the workspace.

The instance of the hierarchy object in the data flow has options you can set to specify:

- A single controlling area or all controlling areas from which the hierarchy data is extracted.
  If you imported the hierarchy type from a controlling area in the browser, or if you specified a controlling area when importing the hierarchy by name, a single controlling area is already specified in the hierarchy definition. If you imported the hierarchy type from the All Hierarchies browser section or did not specify a controlling area, you can specify the controlling area for each instance of the hierarchy.
  Choose a single controlling area from the list for best performance.
- A single hierarchy or all hierarchies in the hierarchy type.
  Choose a single hierarchy for best performance. Enter the hierarchy name as it is shown in the external browser of the datastore.

12.4.4.2.3.3 Example data flow with an SAP source

This section describes one technique for extracting and transforming the hierarchy so you can easily report on any level of the hierarchy.

The job in this example consists of three data flows:

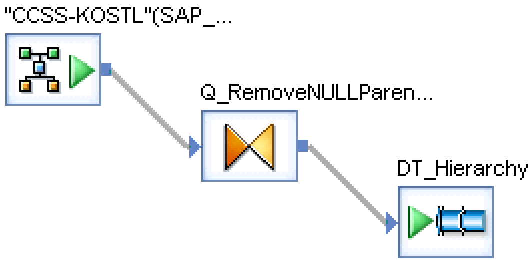
Data flow	Description
Hierarchy_Extraction	An ABAP data flow that extracts the Business Areas hierarchy for Controlling Area 1000.
Hierarchy_Leaf_Values	An ABAP data flow that associates the leaf data with the child nodes of the hierarchy.

Data flow	Description
DF_Hierarchy_Flattening	A data flow that horizontally flattens the hierarchy.

12.4.4.2.3.3.1 Hierarchy_Extraction data flow

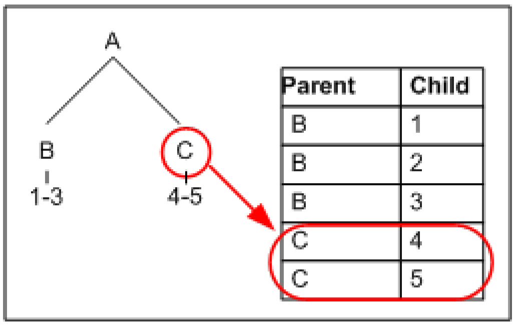
The Hierarchy_Extraction data flow is an ABAP data flow that extracts the hierarchy from SAP. It performs these operations:

1. Extracts the hierarchy type from SAP.
2. Filters NULL parent rows from the output.
3. Maps the parent, child, leaf and root flags, and controlling area columns to the output.



12.4.4.2.3.3.2 Hierarchy_Leaf_Values data flow

The Hierarchy_Leaf_Values data flow is an ABAP data flow that extracts values from the set table that correspond to the leaf nodes in the hierarchy. It produces a new set of parent/child pairs where the new parents are the leaf nodes from the hierarchy and the new children are the values associated with the leaf nodes. The result of this data flow is essentially another level to the hierarchy.

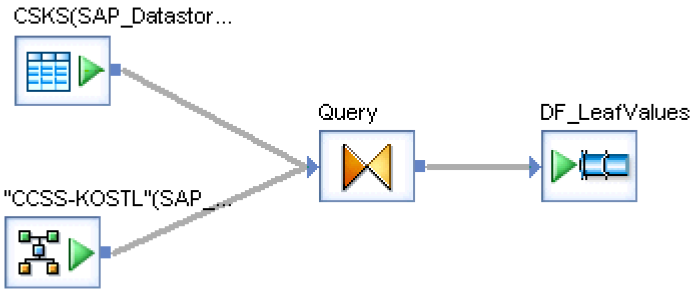


Parent	Child
B	1
B	2
B	3
C	4
C	5

Specifically, the data flow does the following:

1. Joins the hierarchy with the set table to produce a row for each leaf group value, filtering for valid cost centers and Controlling Area 1000.

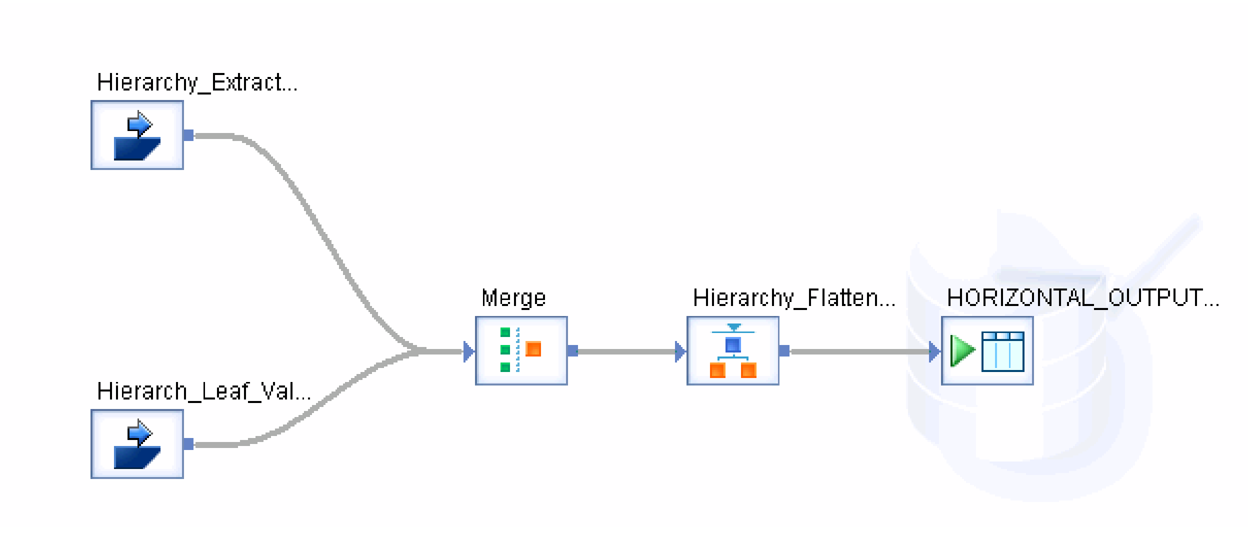
2. Maps the child_id and child_desc from the hierarchy to the parent_id and parent_desc in the output. Maps the cost center id (KOKRS) from the set table to the child_id in the output.
3. Looks up a text description of the cost center id and maps the result to the child_desc.



12.4.4.2.3.3.3 DF_Hierarchy_Flattening data flow

The DF_Hierarchy_Flattening data flow merges the leaf group values with the hierarchy and horizontally flattens the hierarchy as follows:

1. Merges the output from the two ABAP data flows.
2. Horizontally flattens the data using the Hierarchy_Flattening transform.
3. Loads the data into a target.



The output contains a row for each node in the hierarchy including each leaf group value. The table below shows a section of the output including the BA_H2000 cost center group:

Current leaf		Level 0	Level 1	Level 2	Level 3
0HBUS_AREAS	0	0HBUS_AREAS	0HBA_H2000		

Current leaf		Level 0	Level 1	Level 2	Level 3
0HBA_H2000	1	0HBUS_AREAS	0HBA_H2000		
0HH20_3000	2	0HBUS_AREAS	0HBA_H2000	0HH20_3000	
2-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	2-4200
2-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	2-4210
21-3100	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-3100
21-3110	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-3110
21-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-4200
21-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-4210
21-4280	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-4280
0HH20_4000	2	0HBUS_AREAS	0HBA_H2000	0HH20_4000	
2-3100	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-3100
2-3110	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-3110
2-3200	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-3200
2-4100	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4100
2-4120	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4120
2-4130	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4130
2-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4200
2-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4210
21-3100	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-3100
21-3110	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-3110
21-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-4200
21-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-4210
21-4280	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-4280
0HH20_9900	2	0HBUS_AREAS	0HBA_H2000	0HH20_9900	
2-1110	3	0HBUS_AREAS	0HBA_H2000	0HH20_9900	2-1110
2-1200	3	0HBUS_AREAS	0HBA_H2000	0HH20_9900	2-1200
2-1210	3	0HBUS_AREAS	0HBA_H2000	0HH20_9900	2-1210
...		...	...	...	...

12.4.4.2.4 Creating custom ABAP transforms

You can create ABAP programs and incorporate them into an ABAP data flow as custom ABAP transforms. This section gives you information about creating and using custom transforms based on ABAP programs.

12.4.4.2.4.1 ABAP program requirements

You can map a custom-written ABAP FORM logic block to a Data Services custom ABAP transform. The ABAP logic block needs to perform the following functions:

- Extract data out of the SAP application using custom logic that is not currently supported by Data Services ABAP generation logic—for example, import statements and submit statements.

  > Note
  >
  > A subset of call function statements is supported: Functions returning scalar values are supported; ones returning tables are not.

- Load the resulting data into an output schema defined for the custom ABAP transform. You can also define and pass parameters to a custom ABAP transform. To include executing the ABAP logic block as part of your Data Services data movement, define a custom ABAP transform via the transforms tab. Then add it to an existing ABAP data flow.

12.4.4.2.4.1.1 Modifying custom ABAP for Data Services

Use an SAP FORM to modify a custom ABAP program. The custom ABAP must be enclosed in a FORM block to work with Data Services. In the structure shown below, notice the beginning of the FORM section after the comments. Use a FORM to pass the parameters to the output.

```
Comments
Start "FORM" block
Customer's custom ABAP
Start "OTAB1" block
End "OTAB1" block
End "FORM" block
```

12.4.4.2.4.1.2 Custom ABAP program tips

FORMNAME should be entered as it is shown at the beginning of the custom ABAP program. It will be replaced when the final ABAP program is generated.

$CONT_AREA and $ROOTNODE are parameters. Their data types are referenced in the ABAP. These optional parameters can be passed to the transform if defined in the Variables and parameters interface of the Designer and initialized.

After the FORM block, add a OTAB1 logic block immediately after the custom ABAP and before the ENDFORM line. The custom ABAP extracts the results from the SAP application server. The OTAB1 logic block takes these results and moves them into an internal table called OTAB1. You should not declare the OTAB1 in the FORM because it will be replaced in the final generated ABAP. The internal table OTAB1 is replaced by ITAB1 and made accessible to Data Services. Use the OTAB1 logic block as a guide, but it must be modified to follow the logic of your custom ABAP.

12.4.4.2.4.2 Using ABAP logic blocks in transforms

If you have ABAP logic blocks you want to use in your Data Services data movement, you can make them available as transforms.

12.4.4.2.4.2.1 To import an ABAP logic block

1. Open the object library to the *Transforms* tab.
2. Right-click the *Platform* node and choose *New ABAP Transform*.

   A new ABAP transform appears in the object library.
3. Right-click the transform and choose *Properties*.
4. Change the name of the transform so users can find it in the object library.
5. Click *OK*.
6. Right-click the transform and choose *Open* to define the custom ABAP transform.
7. For *ABAP language file name*, enter the full path name of the file containing the ABAP logic block.

   You can also click the button next to the box to navigate to the file.
8. For *Join rank*, you can enter a weight to indicate the rank of this data set in a join. You can also set this value when the transform is used in an ABAP data flow.

12.4.4.2.4.2.2 To define the output schema created by the transform

1. In the output schema, right-click and choose *New Output Column*.

   A Column Properties window opens.
2. Enter the *Name* of the column.
3. Select the *Data type* of the column.
4. Enter the *Content type* of the column.
5. Enter a *Description* of the column.
6. Click *OK*.
7. Repeat for each column in the schema produced by the ABAP logic block.

12.4.4.2.4.2.3 To define the input and output parameters for the transform

1. From the *Tools* menu, choose the *Variables* option to open the Variables and Parameters window.

   The transform is the current context in the window.

2. Right-click in the window and choose *Insert*.
3. Right-click the new parameter and choose *Properties*.
4. Change the name of the parameter.

   Choose a *Name* that identifies this parameter so users can map appropriate data to it.
5. Set the *Data type* of the parameter.
6. Set the *Parameter type*.
7. Repeat to define additional parameters.

12.4.4.2.4.2.4 To use the transform in an ABAP data flow

1. Open an ABAP data flow in the workspace.
2. Drag the transform from the object library into the workspace.

   The transform appears in the ABAP data flow as a source.
3. Construct the rest of the ABAP data flow using the data schema produced by the transform.

12.4.4.2.4.2.5 To pass values to the transform

1. With the ABAP data flow open in the workspace, click ▶ *Tools* ▶ *Variables* ◀.

   The *Variables and Parameters* window opens with the ABAP data flow as its context.
2. Click the *Calls* tab.
3. To set a value for one or more parameter(s), double-click each parameter and specify an expression or value in the *Value* field.

12.4.4.2.4.3 Troubleshooting ABAP programs

Custom ABAP functions can cause a FUNCTION_CONFLICT runtime error executing generated ABAP if a parameter data type specified in the custom ABAP is not the same as expected by the SAP application.

For example, you write a custom ABAP program that writes a varchar(10) value to an SAP system variable, but SAP expects a char(30) value for the same variable. When you execute a job that contains a call to the custom ABAP program, the job fails at run time and adds a function conflict error to the error log for the execution.

12.4.4.2.5 Optimizing ABAP

Data Services generates ABAP code to extract data from SAP applications. Data Services optimizes the generated code to make the best use of SAP application resources. When optimized, the generated ABAP program executes as fast as possible with minimal impact on SAP applications. By producing the best possible ABAP program, Data Services minimizes execution time and maximizes total throughput.

Data Services makes use of the features offered by different version of the SAP platform. Therefore, the ABAP code Data Services generates depends on the version you are running.

12.4.4.2.5.1 Optimizations using SAP application open SQL features

Data Services optimizes ABAP using the open SQL features used by different versions of SAP applications. The open SQL features include:

- Inner joins
- Outer joins
- Aggregate functions
- Group by clause
- Order by clause
- Distinct statement

12.4.4.2.5.1.1 SQL features available in SAP applications

All supported versions of SAP applications support these open SQL features:

- Aggregate functions (`AVG, MAX, MIN, SUM, COUNT*`)
- Group by clause
- Order by clause
- Distinct statement
- Inner joins
- Outer joins

When possible, Data Services generates ABAP code that pushes these operations to the SAP database. To push the operation to the database, several conditions must be met:

- Query can be completed by a single ABAP `SELECT` statement
- Select list does not contain `count()` aggregate.
- No expression is inside an `aggregate` function.
- Parameter column of the aggregate function is a Data Services numeric data type.

> Note
>
> Data Services numeric data types include integer, real, and double. The SAP NUMERIC data type is mapped to Data Services type VARCHAR, which is not a numeric data type.

When any of these conditions is not met, Data Services generates ABAP code that completes the operation within ABAP.

12.4.4.2.5.1.2 Optimizing inner joins

SAP applications can retrieve related data from multiple tables with a single SELECT statement using the JOIN command. You no longer need to use nested SELECT loops. Because a single SELECT statement requires fewer database accesses, performance improves.

When you create ABAP data flows that join transparent tables and that use at least one simple join predicate in the ON clause, Data Services creates a single SELECT statement using the JOIN command and pushes the operation to the SAP application database layer. For inner joins, simple join predicates are AND, =, <>, <, >, <=, and >=. These predicates must evaluate simple columns. Any complex join predicates are evaluated inside the select loop.

Data Services uses the table's join rank to determine which tables to join into a single SELECT statement and the order to join the tables. Specifically, tables that have the same join rank are joined together with a single SELECT statement that is pushed to SAP. Tables with higher join ranks are joined before tables with lower join ranks. Tables or groups of tables with different join ranks are joined with nested SELECT loops. These joins are not pushed to SAP.

When a table has the default join rank (zero), Data Services determines the best join rank for that table. Data Services decides based on which fields the table shares with other tables, using the fields that link the tables in the WHERE clause. Other tables in the join keep their assigned join rank.

When joining tables with a single SELECT statement, SAP determines which is the inner table and which is the outer table of the join. When joining tables with nested SELECT loops, Data Services uses the table or joined tables with the higher join rank as the outer table and the table or joined tables with the lower join rank as the inner table.

For example, suppose you want to join five transparent tables, A through E, with three different join ranks.

Table	Join Rank
A	10
B	10
C	5
D	3
E	3

Data Services creates a single SELECT statement joining A and B and pushes the operation to SAP (these tables have the same join rank and the highest value).

Next, Data Services creates a single select statement joining D and E and pushes the operation to SAP (these tables have the same join rank but a lower value). In both cases, SAP determines which table is the inner table and which table is the outer table.

Finally, Data Services creates a nested SELECT loop that joins the joined tables A and B with table C and joins the result with the joined tables D and E. The joined tables A and B are used as the outer table in the first join, and the joined tables A, B, and C are used as the outer table in the second join (the table with the higher rank is used as the outer table).

Now, suppose you specified four different ranks.

Table	Join Rank
A	10

Table	Join Rank
B	4
C	5
D	3
E	3

In this case, Data Services creates a single SELECT statement joining D and E, and pushes the operation to SAP. Next, Data Services creates a nested SELECT loop that joins table A with table C, joins the result with table B, and joins that result with the joined tables D and E. Table A is the outer table in the first join, the joined tables A and C are the outer tables in the second join, and the joined tables A, C, and B are the outer tables in the third join.

Finally, suppose you specified five different ranks for the five tables.

Table	Join Rank
A	10
B	4
C	5
D	3
E	8

In this case, none of the joins are done with a single SELECT statement that is pushed to SAP. Instead, Data Services creates a nested SELECT loop. The loop joins A with E, the result with C, that result with B, and that result with D.

When Data Services determines which table is the outer table and which table is the inner table of a join, it uses information about the table size and estimates the effect any filters will have on the number of rows retrieved. Data Services uses the smaller data set as the outer table in a join.

Data Services determines the table size based on the *Estimated row count* attribute for the table. You can enter a value for any table.

For joins containing nested relations, Data Services uses the statistics of the embedding table when determining the best join order. For joins containing clustered fields, Data Services uses the statistics of the transparent table when determining the best join order.

Related Information

Specifying sources [page 2454]
Optimizations using nested SELECT statements [page 2469]
Determining the best join order [page 2471]
Reference Guide: Table [page 956]

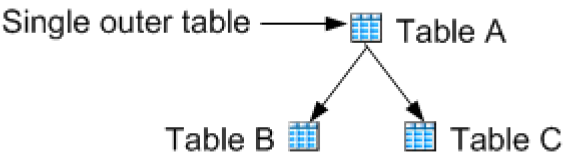
12.4.4.2.5.1.3 Optimizing outer joins

SAP applications can perform an outer join between two tables using the LEFT OUTER JOIN command. Using this ABAP command avoids awkward code that leads to unnecessary database table scans and poor performance.

SAP can only use the LEFT OUTER JOIN command when joining transparent tables with at least one simple expression in the ON clause. For left outer joins, simple expressions specify fields using AND commands with an equals condition. If the join also specifies a complex expression, Data Services will evaluate that expression inside the SELECT loop. Furthermore, SAP applications only support left outer joins when one table is used as the outer table in the join. In other words, within SAP you cannot have an outer join between two tables and use the result in another outer join.

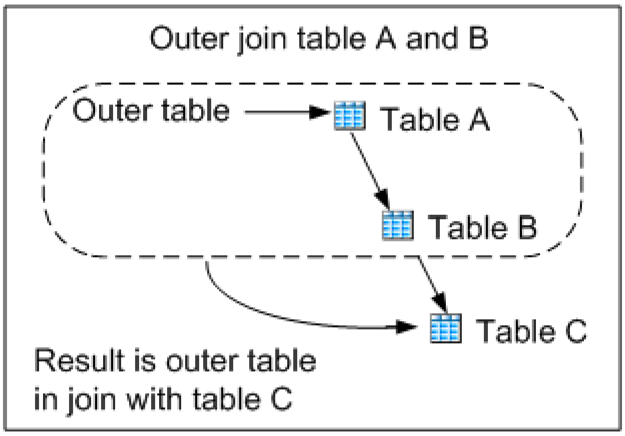
When you create ABAP data flows with an outer join that joins transparent tables and uses simple expressions in the ON clause, Data Services attempts to use the LEFT OUTER JOIN command and push the operation to the SAP application database layer.

When the outer join only uses one table as the outer table in the join, Data Services can create a single SELECT statement using the LEFT OUTER JOIN command and push the operation to the SAP application database layer.



You can use the join rank to control the order in which tables are joined.

When the outer join uses more than one table as the outer table in the join (such as the result of a previous outer join), then Data Services cannot create a single SELECT statement. Instead, Data Services chooses one outer join to complete using the LEFT OUTER JOIN command and then uses a nested SELECT loop to complete the join. Data Services randomly chooses which tables to join using the LEFT OUTER JOIN command.



12.4.4.2.5.2 Optimizations using nested SELECT statements

When working with SAP R/3 version 4.0 or later, you might need to use a nested SELECT statement because of restrictions on using the JOIN command and WHERE clause. When joining tables using a nested select statement, Data Services creates ABAP code that joins the tables in the order it anticipates is most efficient. You can override the decision with manual settings.

When joining data in a nested SELECT statement, Data Services extracts data in loops—the first table is read in an outer loop and subsequent tables are read in an inner loop. The number of loops depends on the number of tables joined. For best performance, you want the table that limits the most records read in the outer loop. By defining the smaller data set first, you will limit database operations when searching for matching records in the inner table. Data Services uses rules to ensure that this happens. You can override these rules and manually specify the join order.

During a join, only two tables are read at one time. However, outer joins that are split into multiple query transforms can reduce extraction performance. Therefore, join performance is better if outer joins that include more than two tables are specified in a single query transform.

Related Information

Designer Guide: Query transform overview [page 310]

12.4.4.2.5.2.1 Automatically determining join order

Data Services automatically identifies the most appropriate table to read first in the join (that is, the table to use as the outer table). Data Services applies several rules:

1. Use the smaller source.
   Data Services estimates the table size using the Estimated_Row_Count attribute for the table. You can change this value to better reflect the actual size of the data set.
2. Use the table with fewer key fields specified in an ON clause.
   When joining tables, the ON clause specifies key fields that must match. Some tables, such as a detail table that is joined with a header table, have additional key fields that are not used in the match. Typically, the table with additional key fields is larger; the table with fewer key fields defines a smaller data set.
   Data Services detects the number of keys, and uses the table with fewer key fields in the outer SELECT loop and the table with the extra keys in the inner SELECT loop.

   > ℹ Note
   >
   > It is important to use all the SAP-specified keys when joining two related tables. If they are not included in the join, the underlying database used by the SAP application may scan the entire table rather than use the primary index.

3. Use the table with a constraint in the WHERE clause.
   When the WHERE clause in the joining Query transform includes a constraint, Data Services assumes that the selection based on a literal constraint will result in fewer selected records and that this table should be in the outer SELECT loop (which may not be the desired result).

For joins containing nested relations, Data Services uses the statistics of the embedding table when determining the best join order. For joins containing clustered fields, Data Services uses the statistics of the transparent table when determining the best join order.

Related Information

Reference Guide: Table [page 956]

12.4.4.2.5.2.2 Manually setting join order

In some cases, you may not want Data Services to automatically determine the join order. For a particular join, the ON conditions or keys may lead to less-than-optimal performance when Data Services applies its rules to determine join order. In these cases, you can specify the join order manually.

To specify the join order manually, set the join rank to a positive non-zero integer for each table in the join. Set the join rank using the Query editor. The table with the largest join rank becomes the outer table in the join. The other tables are included based on the order of their join ranks.

If the join ranks of all source tables remain at the default value (0) and are also not specified in the Query editor, Data Services automatically assigns the join rank.

Even though the join ranks specify tables in the proper order, the underlying SAP application access methods might still prevent optimal performance. In this case, consider alternatives to the multiple table join, including using a two-way join with a table lookup using caching or small tables.

Related Information

Specifying sources [page 2454]
Automatically determining join order [page 2470]

12.4.4.2.5.2.3 Determining the best join order

Use the information available for the data sets involved in a join to determine the best join order. This section shows two types of common joins and the techniques for determining the appropriate join orders.

12.4.4.2.5.2.3.1 Header and detail tables

For “header” and “detail” table joins, you usually want the header table to be the outer table and the detail table to be the inner table.

Because there are typically many line items for each header, the header table defines the smaller data set. Therefore, you want to use the header table as the outer table in the join and the line item table as the inner table in the join. The join operation reads all of the rows of the detail table into memory. Then for each row of the header table, the join matches multiple rows from the detail table. Producing the output requires reading through each table one time, with comparisons made for each row in the smaller table.

The alternative would be to read the header table first, then for each row of the detail table, read through the header table to find the matching details. Producing the output in this alternative requires reading through each table one time, with comparisons made for each row in the larger table. The cost of the join operation is higher in this case.

For example, when extracting rows from a purchase order line item table, you would join them with the corresponding row from the purchase order header table.

This join would produce a row for each controlling area and document number identified in the header table. You might limit the results by setting bounds on when the document was created, using document creation dates from the header table.

12.4.4.2.5.2.3.2 Identifier and description tables

When joining tables to extract text "“descriptions”" for “identifiers”, you often find that the description table is much larger than the identifier table. In that case, you want to use the identifier table as the outer table and the description table as the inner table.

The join operation reads all of the rows of the description table into memory. Then for each row of the identifier table, the join matches a row from the description table. Producing the output requires reading through each table one time, with comparisons made for each row in the smaller table.

For example, when extracting business areas and descriptions, you might join areas from the business area table (the identifier) with text descriptions from the business areas description table (the description).

Because the description table is larger than the identifier table, you want the join to use the identifier table, TGSB, as the outer table and the description table, TGSBT, as the inner table. This join would produce a row for each business area identifier in the TGSB table. You might limit the results by language values from the TGSBT table.

12.4.4.2.5.2.4 Checking join order in ABAP

Before actually running a job, you can check the order in which tables will be joined by examining the ABAP that Data Services generates for the ABAP data flow. In the ABAP, look for the appropriate SELECT loop description. The outer table is described before the inner table.

For example, for a join between two tables MARA with join rank 2 and MAKT with join rank 1, the ABAP contains two separate SELECT statements for the outer table (MARA) and the inner table (MAKT). The generated *ABAP Language* window might appear as follows:

```
select
  MATNR
  MTART
  MBRSH
  MATKL
into corresponding fields of MARA from MARA.
ALTMP2 = MARA-MATNR.
ALTMP3 = MARA-MTART.
ALTMP4 = MARA-MBRSH.
ALTMP5 = MARA-MATKL.
select
  MATNR
  SPRAS
  MAKTX
into corresponding fields of MAKT from MAKT where
((SPRAS = 'E')
AND MATNR = MARA-MATNR))
ALTMP1 = MAKT-MAKTX.
  MOVE ALTMP1 TO ITAB3-MAKTX.
```

where :

- The first SELECT statement is for the outer join, followed by the column list from the outer table
- The second SELECT statement is for the inner join, followed by the column list from the inner table
- The join expression in the WHERE clause displays at the end

12.4.4.2.5.2.4.1 To generate and view ABAP

1. Open the ABAP data flow in the workspace.
2. Choose ▶ *Debug* ▶ *Generate ABAP* ◀.

   The ABAP Language window opens in the workspace.

   If there are errors in the ABAP data flow, the Output window opens with a list of any errors or warnings. There cannot be any errors in the data flow for ABAP to be generated.
3. Locate the correct SELECT statements.

   Use the Find button or browse through the code by scrolling.

The first table in the set is the outermost table.

12.4.4.2.5.3 Optimizations for table caching

SAP applications can cache a table (that is, load it into memory). You can cache tables in two cases:

- Tables used in a join
- Tables used in a lookup function

12.4.4.2.5.3.1 Tables used in a join

Caching tables used as inner tables in joins improves performance as fewer database accesses are required. You can manually set or override table caching.

> ℹ Note
>
> Best practice when joining multiple sources is to set join rank and cache for all sources in the Query editor.

When sources are joined using the Query transform, the cache setting in the Query transform takes precedence over the setting in the source. The default option in the Query editor is Automatic and in a table source is No. The automatic setting in the Query editor carries forward the setting from the source table. The following table shows the relationship between cache settings in the source, Query editor, and whether the software will load the data in the source table into cache.

Cache Setting in Source	Cache Setting in Query Editor	Effective Cache Setting in ABAP data flow
Yes	Automatic	Yes
No	Automatic	No
Automatic (Data Services determines whether or not to cache the table based on table size and number of filtered records)	Automatic	Automatic (based on table size and number of filtered records)
Yes	Yes	Yes
No	Yes	Yes
Automatic	Yes	Yes
Yes	No	No
No	No	No
Automatic	No	No

Note

If any one input schema has a cache specified in the Query editor, the Data Services Optimizer considers only Query editor cache settings and ignores all source editor cache settings.

The default option in the Query editor is Automatic and in a table source is No.

If you enable the cache on all tables in a join, inner and outer tables are read into internal tables prior to a join. This construct replaces SELECT LOOP processing. You should then use the Package Size option on the outer loop of the join to limit the size of the internal tables brought into memory.

The rows extracted from cached tables must fit into the production SAP system memory. When you select the *Automatic* option in the table source, Data Services estimates the table size and number of filtered records and projects the number of rows that must be extracted. When Data Services projects that 2,500 or fewer rows will be extracted and the table is used as the inner loop of a join, Data Services caches the table. To estimate the table size, Data Services uses the *Estimated row count* attribute for the table. You can enter a value for any table.

When cached tables do not fit into memory, Data Services throws an overflow or memory contention error when executing the ABAP program. Errors can occur when tables you select for caching (tables with the cache option set to *Yes*) do not fit into memory or when tables Data Services selects for caching (tables with the cache option set to *Automatic*) do not fit into memory.

If you experience either type of error, you have two options:

- Change the cache option to *No* for one or more tables, and re-run the program until you can execute it without an error.
- Use the Package size option to limit the number of rows read into memory.

Related Information

Specifying sources [page 2454]
Reference Guide: Table [page 956]

12.4.4.2.5.3.2 Tables used in a lookup function

Caching the values that a lookup function needs from a translation table reduces the number of times the system must be queried and improves system performance.

When creating a lookup function, you can specify whether or not to cache the translation table. When you specify that a translation table is cached, Data Services is able to improve performance. There are three options:

Option	Description
NO_CACHE	Data Services caches no values.
PRE_LOAD_CACHE	Data Services applies filters and caches all the matching lookup columns in the translation table. Select this option if the table can fit into memory.
DEMAND_LOAD_CACHE	: Data Services searches the ABAP data flow for other lookup functions that use the same translation table and have the same cache setting. When Data Services finds two or more lookups, it combines those lookups and caches the results into an internal table. The SAP application reads the internal table to find the results of individual lookups. Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.

Because a lookup may retrieve many records, you must be careful that the caching requirement does not exceed available memory. If you get an error message indicating a lack of memory, you should remove caching from one or more tables.

Related Information

lookup [page 2572]

12.4.4.2.5.4 Making table size information available to jobs

When optimizing execution algorithms, Data Services makes decisions based on the size of tables being imported.

You can:

- Make table size information known to jobs for automatic tuning
- Use cost coefficients to further tune job performance

12.4.4.2.5.5 Optimizations for testing ABAP data flows

You can control the number of rows returned from an ABAP program when you run its job in test mode.

When you drop an ABAP data flow into the workspace, the Properties window for that data flow appears. On the *Options* tab, use *ABAP Row Limit* to limit the number of rows that the data flow reads. Then, select the *Test Mode* option from the *Execution Properties* window when you execute this job. This feature allows you to quickly run jobs in a development environment using SAP application data.

Related Information

Adding an ABAP data flow [page 2452]

12.4.4.3 Executing batch jobs that contain ABAP data flows

This section contains specific information about executing batch jobs that contain ABAP data flows.

> Note
>
> This section assumes you are an experienced SAP user and have already followed similar procedures. Your system may be configured differently than the examples, so you should treat the steps in the procedures as high-level information—you may need to fill in additional details or you may need to do the steps in a somewhat different order.

Related Information

Designer Guide: Executing Jobs [page 421]

12.4.4.3.1 Data Services functions supporting ABAP data flows

Using ABAP data flows, Data Services extracts data from SAP applications by directly accessing the SAP application server with Data Services-generated ABAP.

Data Services can execute the ABAP by the following methods:

- *Generate and Execute*
  This method calls the Data Services function /BODS/RFC_ABAP_INSTALL_AND_RUN. If this Data Services function was not uploaded to the SAP server, then the SAP function RFC_ABAP_INSTALL_AND_RUN is called.

  > i Note
  >
  > This method requires that users who submit the job have S_DEVELOP authorization. Most SAP production systems do not provide this combination of authorizations to one user. Additionally, the RFC_ABAP_INSTALL_AND_RUN function is not well suited to execute in parallel. When multiple users access the system at the same time, some jobs could fail due to ABAP program name conflicts.

- *Execute Preloaded*
  This method uses Data Services-provided functions to overcome the security and stability issues inherent in the Generate and Execute method.

Related Information

ABAP execution options [page 2479]

12.4.4.3.2 Validating, generating, and installing ABAP code

Data Services enables you to check generated ABAP syntax errors before you run a job. The errors Data Services detects include mismatches between strings and numerals in parameter passing.

To request ABAP validation, you must have the Data Services function module /BODS/SYNTAX_CHECK installed on your SAP system.

Related Information

Installing Functions on the SAP Server [page 2408]

12.4.4.3.2.1 To validate ABAP code created by Data Services

1. From the Designer, select an SAP data flow object (data flow, table, or transport) in the project area or open an object in the workspace.
2. Select *Debug* > *Validate* > *Validate ABAP* .

   Data Services checks for ABAP errors and opens a window to indicate what the errors are, or that there are no errors.

12.4.4.3.2.2 To generate and install ABAP independent of execution

To install generated ABAP code, you must first meet the following prerequisites:

- You must have necessary privileges in the SAP system to create a new program using transaction SE38.
- If the upload is performed while the SAP Applications datastore is in generate_and_execute mode, you must ensure that SAP client software (SAPGUI) is installed on the machine where the Data Services Designer is running.

> ⚠ Caution
>
> If a program already exists in the SAP system with the same name as an ABAP program you upload with this procedure, the uploaded program overwrites (without confirmation or warning) the previously existing program. You must ensure that name conflicts do not exist.

1. In the local object library, right-click an ABAP data flow object.

   > i Note
   >
   > You can select more than one ABAP data flow object (even from different datastores) and proceed with these steps. However, you won't see the program of any of the data flows, as you do when you work with one data flow at a time.

2. From the context-sensitive menu, select *Generate ABAP Code*.
   The ABAP Programs Generation Dialog appears.
3. In the *Generate ABAP programs in directory* field, specify the path of the ABAP programs.
4. To enable automatic installation of the ABAP programs, select the *Upload generated programs* box.
5. Click *OK*.
   Data Services generates and installs the ABAP programs. Because the SAP system must process the ABAP programs, the installation time could be somewhat lengthy.

12.4.4.3.2.3 To change the Generated ABAP directory

1. In the object library, select an SAP Applications datastore.
2. Right-click and select *Edit*.
3. Click *Advanced*.
4. Enter the new directory path in the *Generated ABAP directory* box.
5. Click *OK*.

12.4.4.3.3 ABAP job execution scenarios

Much of the SAP Applications datastore definition determines how the ABAP data flow portion of a Data Services batch job executes and returns data.

12.4.4.3.3.1 ABAP execution options

For executing ABAP, Data Services provides the following options:

Option	Description
Generate and Execute	ABAP is sent to the SAP application server via the Data Services function /BODS/RFC_ABAP_INSTALL_AND_RUN. If this Data Services function does not exist, then the SAP-supplied RFC_ABAP_INSTALL_AND_RUN is used. This option is designed to be used during the development stage of a project. When a job is executed using an SAP Applications datastore set to the generate_and_execute mode, Data Services generates a random ABAP program name. Use the Designer's ▶ *Tools* ▶ *Options* ▶ *SAP* ▶ *Environment* ◀ option called *Prefix to ABAP program name* to assign a prefix to the random name generated by Data Services. Generated files are stored in the directory you defined for the datastore.
Execute Preloaded	Communication between Data Services and SAP is handled by the Data Services RFC function modules, which must have been pre-installed on SAP. This option is generally used only in the test and production stages of an application project. When you select *Execute Preloaded*, Data Services doesn't generate ABAP to run the job. It simply calls an ABAP program name on the SAP application server. This program was created when you ran the job in Generate and Execute mode. Load the previously generated ABAP program into SAP, rename it as desired, and from the *Options* tab of the data flow *Properties* window, enter its name in the *ABAP program name* box.

12.4.4.3.3.2 Data transfer methods

ABAP data flows in Data Services allow you to automatically create ABAP programs. After an ABAP program runs, it creates a flat file (transport file) on the SAP application host. This file is later transferred from the SAP application host to a directory local to the job server executing the Data Services job.

To use a data transfer method for this outbound SAP file, select an option when configuring a datastore for your SAP application host. Data transfer options specified in an SAP Applications datastore editor determine the method by which the results (from batch jobs that contain ABAP data flows) are returned to Data Services. The options are:

Option	Description
RFC	Use to stream data from the source SAP system directly to the Data Services data flow process using RFC.
Direct download	SAP application server transfers the data directly to the *Client Download directory*.
Shared directory	SAP application server loads the *Working Directory on SAP server* with the transport file. The file is read using the *Data Services Path to the shared directory*.

Option	Description
FTP	SAP application server loads the *Working Directory on SAP server* with the transport file. The file is read using the *FTP relative path to the SAP working directory* and written to the *Client download directory*.
Custom Transfer	SAP application server loads the *Working Directory on SAP server* with the transport file. The file is read by a third-party file transfer program and loaded to the *Local directory*.

12.4.4.3.3.2.1 Custom Transfer method

Using the custom transfer method, you specify a third-party file transfer program which Data Services uses to transport data between the remote SAP application server and a local directory on a Data Services Job Server machine. The transfer from SAP applications to Data Services is enabled using a data transport object (used in ABAP data flows).

When you select the *Custom Transfer* option, provide:

- A transfer program (to be invoked during job execution)
- (optional) Additional arguments used by the transfer program such as security, compression, or Data Services system variables.

Like other datastore settings, custom transfer program settings can be overridden if they are changed in a particular ABAP data flow. You can also edit the *Custom Transfer* option when exporting an SAP Applications datastore.

Datastore profiles support the *Custom Transfer* option.

Related Information

Designer Guide: File transfers [page 269]

Designer Guide: Creating and managing multiple datastore configurations [page 239]

12.4.4.3.3.2.1.1 Data Services system variables to transfer a file from SAP applications

System variables can be used to provide the connection data you enter into Data Services to the custom transfer program. The following system variables can be used in the *Custom Transfer Arguments* box:

Data entered in this field:	Is substituted for this variable if it is defined in the Arguments box
SAP application server	$AW_SAP_HOST

Data entered in this field:	Is substituted for this variable if it is defined in the Arguments box
Working directory on SAP server	$AW_SAP_WORK_DIR
Custom transfer user name	$AW_USER
Custom transfer password	$AW_PASSWORD
Local directory (Local denotes the job server computer)	$AW_LOCAL_DIR
File name This field is on the data transport object and is part of an ABAP data flow.	$AW_FILE_NAME

12.4.4.3.3.2.1.2 Shared directory requirements

To use the Shared Directory data transfer method, set up a shared directory between the SAP application server and your Data Services Job Server computer:

- If the SAP application server is on UNIX, use NFS or a similar disk sharing process. Use these steps to verify that the Job Server can access files generated by SAP:
  a) Log in to the SAP application server computer.
  b) Sign onto the NFS system.
  c) Navigate to the directory you specified in the datastore as the "Working directory on SAP server."
  d) Enter the command `touch bodi`.

     Note that you can use any term instead of bodi. The touch command creates a file with zero length in the current directory.
  e) Log into the Data Services Job Server computer.
  f) Navigate to the directory you specified in the datastore as the "Data Services path to the shared directory."
  g) Look for the file `bodi`.
  h) Use the command `file bodi` to verify that the Job Server can read the file.
- If the SAP application server is on Windows, use Windows directory sharing to provide mutual access to the staging area. Include paths to:
  - The shared device on the application server
  - The shared device on the Data Services Job Server

> **Note**
>
> Because both the Data Services and the SAP application server can run on either Windows or UNIX platforms, mapping between computers can be interrupted if one of the computers is rebooted.

Observe the following recommendations:

- If you are running SAP and the Data Services on UNIX, make sure that your network will automatically mount the volumes you want to use when any computer is re-booted.
- If you are running the SAP application server on Windows and the Data Services Job Server on UNIX and you re-boot the UNIX computer, you may need to re-map Data Services to the SAP application server. If your jobs return error messages such as the following, disconnect then remap the NFS connection in SAP.

  ```
  The SAP Job Canceled Host...
            File xxx cannot be opened...
  ```

Related Information

ABAP job execution scenarios [page 2478]

12.4.4.3.3.2.2 Direct download transport method

With *Direct download*, the data produced by the SAP application after running the ABAP program is returned by the Job Server computer via a ws_download function call. The data is stored in the *Client download directory*.

You should avoid using this method except when you need a quick way to return a small amount of data; it might not be reliable and does not scale well.

12.4.4.3.3.2.3 FTP transport method

The FTP method is most useful in heterogeneous environments (Windows and UNIX). It scales well and provides a good level of security, making it a recommended method of data transport.

12.4.4.3.3.2.4 Shared directory transport method

With shared directory transport method, both Data Services and SAP applications have direct access to the directory where data is stored. This method works best in a homogeneous environment (both systems are on Windows). The security risk inherent in a shared directory may pose a problem for some organizations. If this is the case, FTP is probably a better choice.

12.4.4.3.4 Sample execution sequence for an ABAP job

This section describes how a batch job that contains an ABAP data flow (also called an ABAP job) executes.

The following execution sequence is for a hypothetical job with the following characteristics:

- The batch job contains a data flow that, in turn, calls an ABAP data flow.
- The ABAP data flow extracts data from a subset of the columns in an SAP table.
- The calling data flow passes the output from the ABAP data flow into a query transform and loads a target table or file.
- The datastore options selected are *generate_and_execute* and *shared_directory*.

The scenario is valid for jobs initiated in Data Services as well as jobs initiated in SAP.

Oracle is assumed for this example, but the process is similar for other environments.

12.4.4.3.4.1 The execution steps

1. Data Services retrieves the job specification from the repository.
2. Data Services validates the job specification.
3. Data Services performs any required optimizations.
4. Data Services compiles the job specification.
   As part of this step, Data Services generates an ABAP program corresponding to the ABAP data flow specification and writes it to the path specified in the SAP Applications datastore options.
   All the parameters are replaced with data before ABAP is submitted to SAP to execute.
5. Data Services invokes `RFC_ABAP_INSTALL_AND_RUN` with the ABAP program as a parameter.
6. Depending on whether the ABAP job is running in the background or foreground, one of the following happens:
   - If the *Execute in background (batch)* option is set to *No*, then the ABAP program runs as a foreground job in dialog user mode in SAP. The `RFC_ABAP_INSTALL_AND_RUN` return code is checked for status. (`RFC_OK` return status implies successful completion of the ABAP program.)
   - If the *Execute in background (batch)* option is set to *Yes*, then the ABAP program runs in the background in SAP. A special Data Services-internal ABAP program runs in SAP and polls for status from the job status table TBTCO based on the job name that was assigned. The status column TBTCO-STATUS provides status information.
7. The ABAP portion of Data Services SAP-related jobs cannot run in parallel, so Data Services serializes the requested processes.

   > Note
   >
   > ABAP jobs can run in parallel when you specify execute_preloaded and the job executes with Data Services RFC.

8. Data Services schedules the job by calling SAP functions JOB_OPEN and JOB_CLOSE.
9. The job is scheduled as a class C job.
10. Data Services queries the job status every 15 seconds.
11. When the job finishes, the ABAP is deleted, but the job is still listed with a status of "finished."
12. When the ABAP program completes successfully, it writes output to the file name specified in the data transport in the path specified as *Working directory on SAP server* in the datastore definition.
13. The ABAP data flow is done and returns to the calling data flow.
14. Data Services reads the output file generated from ABAP data flow execution and processes it.
    Processing may include performing joins with other ABAP data flows or tables from other source databases or source files. It may also include any column transformations and/or transforms applied to the entire SAP

output data set. Once all transformations are done, Data Services is ready to place the resulting data set in an output file or target table.

15. If the output is to a target table, these are the options available for loading the table:
    - If Oracle is the target database, then bulk loading is the typical choice.
      If bulk loading is turned on, Data Services generates the control file (.CTL) and the input (to the SQL*Loader) data file (generated_filename.dat) in the `C:/Data Services/Log/BulkLoader/` directory or the bulk loader directory specified in the Oracle datastore. It then invokes SQL*Loader on the client side (the system running Data Services) to load the data according to the control file specification.
    - If bulk loading is not used, Data Services builds and sends batch INSERT SQL statements to the Oracle server to execute and load the rows. A batch size option specifies the commit size.

Related Information

Execution modes of ABAP data flows [page 2487]
Debugging and testing ABAP jobs [page 2489]
Performance Optimization Guide: Using Bulk Loading [page 2164]

12.4.4.3.5 Scheduling jobs from SAP

The following is an overview of the execution process for jobs initiated in SAP.

1. When it is time to run a job, SAP sends an extract request to Data Services through the RFC server.
2. Data Services generates ABAP and submits it to SAP.
3. The Data Services batch file initiates execution. The job runs on the job server specified when the job was exported.
4. The ABAP is executed on the SAP application server.
5. The SAP application server generates the specified results.
6. Data Services is notified of the results.

12.4.4.3.5.1 Creating the program that will execute the schedule

You need to create the ABAP program that will initiate communications with the Data Services RFC server to execute the job schedule.

We recommend that you use the model ABAP code provided for you at `Data Services\Admin\R3_Functions\schedule\yschedul.abap` (the procedures in this section assume you do so). You can also create your own program.

12.4.4.3.5.2 To create an ABAP program in SAP applications

1. Run transaction *SE38*.
2. Select *yschedul* in the *Program* drop-down box.
3. Click the *Create* button.
   The *Program Attributes* screen appears.
4. From the *Program Attributes* screen:
   - In the *Title* box, add a title.
   - In the *Type* box, enter *1* (online).
5. Click *Save*.
6. Click the *Source code* button.
7. In the *Create object catalog entry* screen, enter the development class for this job.
   This can be any development class you have created for this purpose. We recommend using zact.
8. Click *Save*.
   An ID number has been assigned to this job (the correction and transport number).
9. Click *Save*.
   The ABAP/4 Editor: *Edit Program* screen appears.
10. Choose *Upload* from the ▶ *Utilities* ▶ *Upload/Download* ▶ menu.
11. From the Upload dialog, navigate to yschedul.abap.
    The default location is `C:\Data Services\Admin\R3_Functions\Schedule`.
12. Click *OK*.
    This returns you to the ABAP/4 Editor: Edit Program screen.
13. Click *Save*.
14. Click the back arrow to get back to the ABAP/4 Editor: Initial Screen.
15. From the ABAP/4 Editor: Initial Screen, click *Execute*.
16. Add the fully qualified path to the batch file containing the job you want to run and the RFC destination name.
17. Click *Save*.
18. Enter a variant name and (optionally) a description.
    A variant name is a collection of parameters that determine how a job is run. Use any name that is meaningful to you.
19. Click or press *Enter*.
    The SE38 procedure is complete.

12.4.4.3.5.3 To create a job schedule in SAP applications

1. Run transaction *SM36*.
   View the Define Background Job screen.
2. From the Define Background Job screen:
   a) Add a job name.
   b) Add the target host (the SAP application server).
3. Click the *Steps* button.

The Create Step 1 screen appears.

4. In the Create Step 1 screen, click the *ABAP/4* button.
   Items in the ABAP program section are now available.
5. Add the program *Name* (yschedul).
6. Add the appropriate variant information.
7. Click *Save*.
8. Return to the *Define Background Job* screen.
9. Click the *Start date* button.
10. Add the date and time for the job to run.
11. Complete additional information as needed and click *Save* to complete the job schedule.

The job runs at the time you scheduled it.

If Data Services is active when the job begins to run, the execution window (with the trace button active) appears.

12.4.4.3.5.4 To register the Data Services RFC server in SAP applications

1. Run transaction *SM59*.
2. If you have not yet created the TCP/IP connection between Data Services and SAP applications, do so from the TCP/IP connection choice on this screen.
3. When the TCP/IP connection has been created and is active, click the *Create* button on the Display and Maintain RFC Destinations screen.
4. In the RFC destination box, add an RFC destination name.

   This can be any name you choose.
5. Under ▶ *Technical Settings* ▸ *Connection type* ◀, choose *T* (for TCP/IP).
6. (Optional) Under *Description*, add descriptive text.
7. Click or press *Enter*.

   The RFC Destination screen appears.
8. In the *Registration — Program ID* text box, fill in the name of the Program ID.

   This is the job name you specified for RFC Program name in the RFC Server Interface connection (see Starting the RFC server [page 2486]).
9. Click the *Test connection* button to check if the connection is successful.

   If the connection is not successful, you receive an error message. You need to debug the problem before proceeding.

12.4.4.3.5.5 Starting the RFC server

The Data Services RFC (remote function call) server is the execution link for Data Services jobs initiated by SAP applications or SAP NetWeaver BW.

To start the RFC server, in the Administrator from the *RFC Server Interface Status* page, select the desired RFC server and click *Start*. For details, see the *Data Services Administrators Guide*.

You can verify the RFC server is running in several ways:

- In the Administrator, select *SAP Connections* > *RFC Server Interface* > *RFC Server Interface Status* tab.
- View the SAP application processes.
- View the Windows or UNIX processes (depending on where the RFC server has been installed in your environment)
  If the Data Services RFC server is running, it appears among the running processes listed, for example RfcSvr.

Related Information

Management Console Guide: Starting or stopping an RFC server interface connection [page 1936]

12.4.4.3.6 Monitoring ABAP jobs in SAP applications

You can monitor an ABAP job generated by Data Services just as you would any ABAP job using the SM37 batch monitoring window.

The Data Services job name appears in the list of jobs in the SM37 window.

12.4.4.3.7 Execution modes of ABAP data flows

There are two execution modes for ABAP data flows:

- Dialog execution mode
- Batch execution mode

i Note

You cannot use the dialog and batch capabilities at the same time.

Related Information

Dialog execution mode [page 2488]

Batch execution mode [page 2488]

12.4.4.3.7.1 Overview of preparation steps

For jobs where the schedule is set and the job is initiated by SAP, do the following.

1. In Data Services, export the job to a batch file (*.bat file for Windows environments, and *.sh file for UNIX environments).
2. In SAP from transaction SE38, create the ABAP program that will execute the job schedule.
3. From transaction SM36, create the job schedule.
4. From transaction SM59, create the RFC destination by registering the Data Services RFC server with SAP.
5. In Data Services, ensure that the Job Server is running (or will be running when the job begins to execute).
6. Add an RFC Server Interface.
7. Start the RFC server or ensure that it is running (or will be running when the job begins to execute).

In SAP from transaction SM37, you can monitor the job as it runs. You can also monitor the job in Data Services. The Designer does not need to be running. If it is running, the execution window opens when the job begins to run.

Related Information

Management Console Guide: Administrator, Batch Jobs, Using a third-party scheduler (To export a job for scheduling) [page 1900]

Designer Guide: Executing Jobs, Ensuring that the Job Server is running [page 423]

Monitoring ABAP jobs in SAP applications [page 2487]

12.4.4.3.7.2 Dialog execution mode

In dialog execution mode, Data Services connects to SAP, submits and executes an ABAP program as a foreground job in SAP, then disconnects from SAP after the program execution finishes.

Dialog is the default execution mode.

An ABAP program executed in dialog mode is subject to the dialog execution time constraint (typically set to be 5 minutes) defined in the SAP system profile. If the execution time of a program executing in dialog mode exceeds the specified time-out period, SAP aborts execution of the program, producing an ABAP data flow runtime error and an exception error in Data Services.

The execution time constraint is set using the vdisp/max_wprun_time parameter in the profile corresponding to the system executing the job.

12.4.4.3.7.3 Batch execution mode

In batch execution mode, Data Services connects to SAP, submits the ABAP program as a background job in SAP, then disconnects from SAP after the program is submitted. Data Services schedules and executes the submitted program as a background job.

An ABAP program executed in batch mode is not subject to the dialog execution time constraint mentioned in that mode. However, there is a slight performance overhead due to background scheduling in SAP and execution status checking by Data Services.

> Note
>
> If you kill a Data Services job containing an ABAP data flow defined to run in batch mode, the batch program still executes. You must use SAP facilities to terminate the batch program.

When more than one ABAP data flow is running concurrently in batch mode, the batch job submissions are serialized. However, after the batch jobs are submitted, they run in parallel in the SAP application server if the server has more than one batch processor available.

To run the ABAP generated by the ABAP data flow in batch execution mode, click *Advanced* on the Edit Datastore window and set the *Execute in Background (batch)* option to *Yes*. Make sure that the Data Services SAP functions are uploaded to the SAP system running the job.

> Note
>
> The SAP application server name is not case sensitive, but the *Target host* name must be entered as it is actually registered in the SAP system.

Related Information

Installing Functions on the SAP Server [page 2408]

12.4.4.3.8 Debugging and testing ABAP jobs

Generating and executing ABAP programs from Data Services can result in the following kinds of errors:

Task	Type of error	Result
Design	Validation errors	Execution stops. Information is available interactively (in a pop-up error window).
ABAP generation	Generation errors	Execution stops. Information is available interactively (in a pop-up error window).
ABAP validation	ABAP syntax errors	Execution stops. For errors detected by Data Services, detailed information is provided in the error log. For errors detected by SAP, the message in the error log is RFC_CLOSED. In this case, you need to run the ABAP program manually in SAP to identify the problem more precisely.

Task	Type of error	Result
ABAP execution (dialog mode)	ABAP execution errors	Execution stops. Information is available interactively (in a pop-up error window).
ABAP execution (batch mode)	Job execution errors (job was cancelled)	Execution stops. The job log from SAP is available in the Data Services error log.
ABAP execution	Function exceptions	Causes an exception. Information is available in the Data Services error log if the Data Services function modules have been installed on SAP.

Related Information

Installing Functions on the SAP Server [page 2408]

12.4.4.3.9 Generating ABAP code

You can generate ABAP code to be executed manually on SAP.

12.4.4.3.9.1 To generate ABAP code

1. Open the ABAP data flow.
2. Choose ▶ *Validation* ▶ *Generate ABAP code* ◀.

   When the generation is complete, Data Services opens the text file containing the ABAP program.

   This file name and location were specified when you defined the ABAP data flow.

12.4.5 Working with functions

You can use SAP remote function calls (RFCs) in queries created in Data Services data flows. In addition, Data Services provides the SAP application BAPI interface to support the use of remote function calls designed for business transactions (BAPIs). Use the SAP application datastore to import BAPI function metadata.

SAP functions that are not RFC-enabled can be used in ABAP data flows with the following restrictions:

- The function can have only scalar, multiple input parameters. The function cannot use table parameters.
- For the output you can select only one scalar parameter.

Data Services cannot use normal functions in data flows because an SAP application normal function is not an RFC function.

However, you can write a wrapper for a normal function, which can change the function to be RFC enabled. In this case, a normal function would be supported in data flows including table parameters.

Related Information

SAP application RFC and BAPI function calls [page 2568]

12.4.5.1 Calling BAPI functions

A real-time job can apply data to SAP applications using a real-time job remote function call (RFC). The RFCs that are included in SAP application releases, called Business Application Programming Interface (BAPI) function calls (such as BAPI_SALESORDER_CREATEFROMDAT1) can be particularly useful.

The metadata for the function—displayed through the function wizard in a query—describes the input and output parameters of the function, including the columns and tables included in the transaction applied to SAP.

Related Information

Browsing, searching, and importing metadata from SAP applications [page 2440]
Calling RFC functions [page 2491]

12.4.5.2 Calling RFC functions

A real-time job can use RFC-enabled functions to retrieve information from and apply information to SAP applications. With nested tables, you can include tables as input or output parameters to the function. Functions allow you to:

- Return a specific response based on specific input you provide to the function
- Apply data to or retrieve data from more than one SAP table at a time

RFC functions can require input values for some parameters; SAP supplies default values for other inputs, and some can be left unspecified. You must determine the requirements of the function to prepare the appropriate inputs.

> Note
>
> To avoid returning errors from RFC calls, make sure input is formatted as required by SAP. For example, all character data must be in uppercase; some values require padding to fill out the length of the data type.

To make up the input, you can specify the top-level table, top-level columns, and any tables nested one-level down relative to the tables listed in the FROM clause of the context calling the function. If the RFC includes a structure as an input parameter, you must specify the individual columns that make up the structure.

A data flow may contain several steps that call a function, retrieve results, then shape the results into the columns and tables required for a response.

12.4.5.2.1 To call a function that includes tables as input or output parameters

1. Create a table that includes the columns required as input to the function as the input data set to a query.
2. In the output schema of the query, right-click and choose *New Function Call*.
3. In the *Select function* wizard, under *Function categories* select the datastore corresponding to the SAP application server where you want to apply the function.
4. Select the RFC function to call and click *Next*.
5. Enter the input parameters for the function and click *Next*.

   If there are more parameters than fit on a page in the function wizard, click the down arrow to display the next page of input parameters.
6. Select which parameters to return by selecting a column or table to include on the left and moving it to the list on the right. Click *Finish*.
   In the Query, the function appears as a nested table in the output. Your output parameters are listed as columns or tables under the function.
7. The function is automatically marked to unnest in the output of the query, so the nesting level indicated by the function name is not included in the input to the next step in the data flow.

12.4.6 IDoc interface connectivity

Your SAP system should be configured to send and receive IDocs from Data Services. Refer to the SAP application documentation for more information. Set the communication type to Unicode if the SAP system supports it.

IDoc message sources (SAP outbound IDocs) must be configured in the Data Services Designer and in the RFC client branch of the Data Services Administrator. However, you do not need to configure RFC clients for IDoc message targets (SAP inbound IDocs). The partner number, partner type, message type, and IDoc type settings you save in the Designer's IDoc target editors are the ones needed to send messages to SAP application via transactional remote function calls (tRFCs).

The remainder of this section describes Data Services functions and SAP security levels and user authorizations. These sections all apply to the ABAP, RFC, and IDoc interfaces.

Related Information

Source [page 2541]

Importing IDoc metadata using the Search option [page 2441]

12.4.6.1 Batch and real-time processing overview

An SAP IDoc is an intermediate document that can be used to send and receive business documents between SAP application servers and other servers.

SAP applications offer numerous IDocs that describe different business documents. The data in an IDoc supports a particular transaction. An IDoc can contain data from multiple tables and can contain hierarchical data.

Data Services processes outbound and inbound IDocs from and to the SAP application server. SAP outbound IDocs are called IDoc sources in Data Services. SAP inbound IDocs are called IDoc targets.

The method of processing IDocs depends on the type of job in which you define the IDoc and whether the IDoc is handled as a message or a file.

In Data Services, an IDoc can be used as a source or target object in a batch or real-time job. An Idoc can be processed as a message or as a file, however the only limitation is that batch jobs cannot process IDoc messages as sources.

For example, Data Services real-time processes IDoc source messages (outbound IDocs from the SAP application server) as follows:

- The SAP system administrator includes the Access Server as a TCP, remote function call (RFC) destination.
- The SAP application server (client) publishes an IDoc after processing a transaction. For example, when an operator enters a sales order.
- The Access Server receives the published IDoc and dispatches it to an available service provider for processing.
- If no service providers are available for this IDoc type, the Access Server queues the request and dispatches it when a service provider becomes available.
- When available, a service provider processes the IDoc. The IDoc data can trigger queries to the SAP application server or cached data as required to enrich the IDoc data. The real-time job flattens and loads the required information into a data cache and/or an IDoc message target as specified in its data flow design.
- The service provider returns a response to the Access Server indicating the successful IDoc processing.
- The Access Server receives the response and sends an acknowledgment to the SAP application server so the IDoc can be marked as processed.

Related Information

IDoc interface connectivity [page 2492]

Browsing, searching, and importing metadata from SAP applications [page 2440]

IDoc file [page 2562]

IDoc message [page 2564]

Source [page 2541]

Target [page 2548]

12.4.6.2 IDocs and batch jobs

12.4.6.2.1 IDoc sources in batch jobs

In batch job data flows, IDoc file sources can hold one file or many individual files (to be processed at a later time).

IDoc file sources are processed as follows:

- The IDoc file is opened and the IDoc record type is examined to see which type it is.

  > Note
  >
  > IDoc file sources are validated against the IDoc Type, the IDoc Type related hierarchy, and the SAP Release.

- Related IDoc metadata is read.
- The IDoc list (one or many IDocs) is built based on the information in the *IDoc File* box.
  If an error occurs, a message like the following one displays:

  ```
  Cannot build IDoc from IDoc file: <%s>. Please see the error
                  log.
  ```

  The process terminates, and the next file is processed (if multiple IDoc files are defined).
- Each IDoc is read one by one from the IDoc list and passed to the query. A trace message is recorded in the job's trace log such as:

  ```
  Process IDoc number: <no. in the file: filename>.
  ```

Related Information

Source [page 2541]

12.4.6.2.1.1 Multiple file read

You can configure Data Services to read multiple files by listing the names of these IDoc files in the *IDoc file* box of the source editor in the Designer.

Examples:

- If you specify the IDoc file as `D:\temp\IDOC_R3*.txt`, Data Services processes data from all files in the `D:\temp` directory with names beginning with `IDOC_R3` and the extension `.txt`.
- If you specify the IDoc file `D:\temp\IDOC_R3?.txt`, Data Services processes data from all files in the `D:\temp` directory with names beginning with `IDOC_R3`, with any character before the extension, and the extension .txt.
- If you specify the IDoc file `D:\temp\IDOC_01.txt`, `D:\temp\DOC_02.txt`, `D:\temp\IDOC_03.txt`, Data Services processes data from these three IDoc files.

12.4.6.2.1.2 Variable file names

You can use variable file names in the *IDoc File/s* box. Define a global or local variable for an initialization script of a job. For example:

```
$filepath = 'D:\temp\IDOC_R3*.txt'
```

Then enter **`$filepath`** in the *IDoc File/s* box. In this way you can avoid opening each IDoc object to configure the location of source files.

12.4.6.2.2 IDoc targets in batch jobs

In batch job data flows, one or more sources can be transformed into an IDoc message target.

The sources may become active on a scheduled basis. As soon as all sources are present, the transform (a query, for example) processes the data into the IDoc format configured in the query's output schema and the IDoc message(s) are sent to SAP via a tRFC connection.

A data flow that contains an IDoc file target can be used to store IDocs. When you configure an IDoc file target, Data Services allows you to capture data for more than one IDoc. Each time a root segment passes into a file target (for example, *IDoc File* C:\temp\idoc), a new file is created and named using an eight-character suffix starting with `00000000.txt`. For example, an *IDoc File* set to `C:\temp\idoc` can store `idoc00000000.txt` to `idoc99999999.txt`.

Related Information

Target [page 2548]

12.4.6.2.3 Adding an IDoc to a batch job

When you drag an IDoc into a data flow, Data Services displays a popup menu, with valid options: *Make IDoc file source*, *Make IDoc file target*, *Make IDoc message target*.

To add an IDoc to a batch job:

1. Select an IDoc from the object library.
2. Drag it into an open data flow.
3. Select the appropriate option.

12.4.6.3 IDocs and real-time jobs

12.4.6.3.1 IDoc sources in real-time jobs

For real-time data extraction from and loading to SAP applications, Data Services uses SAP Application Link Enabling (ALE) technology and Intermediate Documents (IDocs) to capture and process transactions

Each real-time job can contain only one message source. For extracting data from SAP applications, using Data Services, that message source must be an IDoc.

A real-time job can contain other sources. Data from these supplementary sources is extracted on-demand when the real-time job is processing data from the message source.

The message source object represents the schema of the expected messages. Messages received are compared to the schema. If a data flow receives a message that does not match the schema, the data flow throws an error at run time.

You must import the schema for each SAP IDoc message type your real-time system will process.

Related Information

Defining SAP Applications datastores [page 2438]

12.4.6.3.1.1 Creating real-time jobs using IDoc sources

To create and run a real-time job that uses an IDoc message source, perform the following steps.

First see *IDoc interface connectivity* [page 2492] for information about configuring SAP to send IDocs to Data Services.

In the Designer:

1. Create an SAP Applications datastore.
2. Import IDoc metadata into the Data Services repository.
3. Create a real-time job using one or more data flows.
4. Drag an IDoc Type into the first data flow and select *Make IDoc message source*.

   > **Note**
   >
   > You cannot configure XML message sources or targets in a real-time job that has an IDoc message source.

5. Add a query and a target to the data flow.
6. Connect the objects.
7. Map the schema using the Query editor.
8. Validate and execute the real-time job by running it in test mode. Results are saved to a file you specify.
9. When the job runs as desired, open the Data Services Administrator to set up the job in a test/production environment.

In the Administrator:

1. If you have not already done so, add a connection to the repository that contains the real-time job and add a connection to the Access Server that will manage the real-time service.
2. Add a service and service providers for the real-time job.
3. Configure an Access Server to communicate with SAP to process IDocs in real-time jobs as follows. Under the *Real-Time* node for the Access Server to configure, click *Client Interfaces*. From the *Client Interface Configuration* tab, add an RFC connection, then add supported IDocs from the *List of Supported IDocs* tab. On the *Add IDoc* page, specify the *IDoc type* (used in the job) and the *Service name* (that you just created for the real-time job).
4. From the *Real-Time Services Status* page, start the service for the real-time job.
5. From the *Client Interface Status* page, start the client.
6. From the *Real-Time Services Status* page, monitor the service and view Job Server error and trace logs to verify the service is successfully processing IDocs from SAP.

12.4.6.3.1.2 Request/acknowledge data flow model

The request/acknowledge data flow model supports the use of an IDoc as a message source in the first data flow inside a real-time processing loop. Data Services automatically generates a final data flow in any real-time processing loop that contains an IDoc message source. This final data flow contains an XML message target.

The XML message target sends an acknowledgement back to the Access Server indicating that the IDoc message source was received from the SAP application. The Access Server returns the acknowledgement to the SAP application.

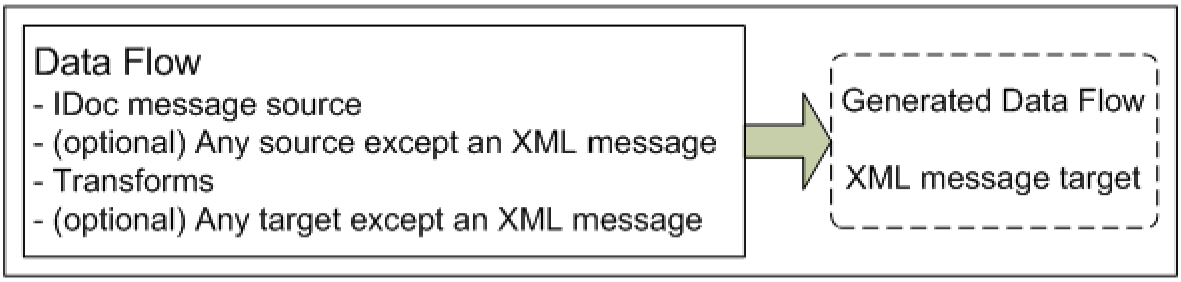

Data Flow
- IDoc message source
- (optional) Any source except an XML message
- Transforms
- (optional) Any target except an XML message
Generated Data Flow
XML message target

Figure 8: Request/Acknowledge model with a single data flow

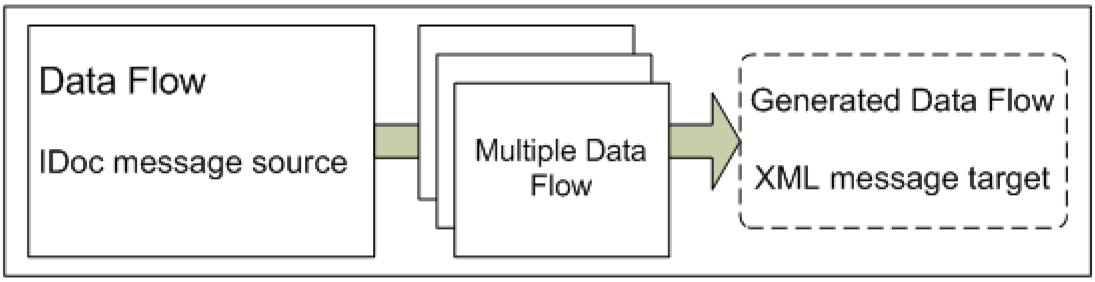
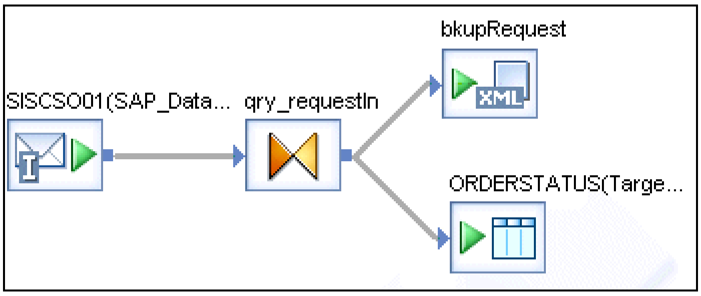


Figure 9: Request/Acknowledge model with multiple data flows

When using the request/acknowledge model, after the first data flow you can configure additional data flows, work flows, scripts, conditionals, and so on. The only limitation is that because an IDoc message source is used, XML message sources and targets cannot be configured in the job. Outside the real-time processing loop you can add initialization and/or clean-up objects as desired.

12.4.6.3.1.2.1 First data flow in a request/acknowledge model

In a request/acknowledge data flow model, the first data flow in a real-time processing loop contains an IDoc message source. To move a data set to succeeding data flows (optional), you can use a memory table as shown in the example below.



12.4.6.3.1.3 Adding an IDoc source to a data flow

Use the following procedure to add an IDoc source to a data flow.

1. Open the data flow in the workspace.
2. In the object library, select the *Datastores* tab.
3. Expand the datastore that defines the connection to the SAP application server.
4. Inside the datastore, expand the IDocs node.
5. Drag the IDoc that the data flow will process into the workspace and choose to use it as a file or message source by selecting *Make IDoc file source* or *Make IDoc message source* from the pop-up menu.

> Note
>
> If your data flow is the first one inside a real-time processing loop, it can have one and only one message source. If another message source already exists in the data flow definition, Data Services will not allow you to place another one in the workspace.

6. Drop a Query and a target (optional if your data flow is part of a real-time job) and connect the objects.
7. Define source editor options by clicking the name of the IDoc to open the source editor.

   If the IDoc source contains nested data, the schema displays nested tables to represent the relationships among the data.

   IDocs contain many nested tables and potentially hundreds of columns across the levels. It is possible that you only need to load data from a fraction of the columns from the IDoc. To make working with the IDoc contents easier, consider deleting columns and nested tables that you do not need.
8. Define the output schema.

   First double-click the Query object to open it.

   You can drag and drop or copy and paste columns from the input schema to the output schema. The icons indicate when a column is mapped. The following table shows sample input and output schemas:

Input schema	Description	Output schema	Description
E1CVBAK-VBELN	Sales document number	E1CVBAK-VBELN	Sales document number
E1CVBAK-AUART	Sales document type	E1CVBAK-E1CVBP0-PARVW	Partner function
E1CVBAK-E1CVBAP-POSNR	Sales document item number		
E1CVBAK-E1CVBP0-PARVW	Partner function		

Considerations:

- All IDoc fields have type char.
- To use two columns with the same name, you must change the name of the first column before adding the second column.

9. (Optional) Join IDoc data with an SAP table using a Query transform.

   Extract data from the SAP application table using an RFC function call. This is efficient for a small amount of data.

Related Information

Source [page 2541]

IDoc file [page 2562]

IDoc message [page 2564]

12.4.6.3.2 IDoc targets in real-time jobs

Most actions that users perform through a Web-enabled application require some response from the system in the form of data. The response can be as simple as an acknowledgment that a message was received. It can also be a specific set of data extracted from an SAP system or data cache. If you have configured a real-time job using an IDoc message source, you do not need to configure a target to send an acknowledgement because Data Services generates an acknowledgement (an XML message) automatically.

IDocs can also be configured as message or file targets in real-time jobs. IDoc message targets move data to SAP applications. IDoc file targets can be used to collect data for testing, data flow validation, and backup.

For example, you might want to create two identical data flows. The first has an IDoc message target and the second has an IDoc file target. The second data flow could be used to store IDocs in a file temporarily. It could be used in a job that your system administrator starts only when it is known that the target server will be down. Later, you can use a job to update the database with the data captured in the IDoc file target.

IDoc target messages can be sent from Data Services on a regular schedule or in real-time because you can choose to use IDoc message targets in either batch or a real-time jobs.

While there can only be one IDoc message source in a real-time job, any number of IDoc targets can be included in a data flow. In addition, if your source is not an SAP application server, a data flow can have an XML message as a source and an IDoc message or file as a target.

Any number of IDoc message targets can be added to a data flow. A data flow can also contain other targets such as a table or file.

Data flows in real-time and batch jobs that contain an IDoc message target send IDocs to SAP applications in the same way. The only difference is when the data is sent. Data can be sent to SAP applications on demand, or if you use an IDoc message target in a batch job, data is sent when the job is scheduled to run.

When a job is executed, output statistics from the IDoc target object are written to the real-time job's monitor log. The number of IDocs written is recorded. The number of IDocs increments for every root-level row of data passed to SAP because IDocs contain only a single root segment.

12.4.6.3.2.1 Data and format considerations

IDoc targets can insert and update data in SAP applications. For example, to create an IDoc target in Data Services, you import metadata from an existing IDoc, then reuse this metadata to create an IDoc target object in Data Services.

12.4.6.3.2.1.1 Data

The following examples list data consistency issues you should consider when planning to use IDoc targets.

Example 1

If your IDoc contains information to update a sales order, then the target IDoc (sales order) must:

- Be defined with a valid IDoc Type (name).
  In SAP, a name for an IDoc is called an IDoc type. IDoc types are subsets of message types. An example of a message type is ORDERSP, while an example of an IDoc type is ORDERS04.
- Match the SAP order number.
- Match SAP segment field formatting.
- Contain all mandatory IDoc segments for its message type.
  SAP segments are sections of IDocs. They are equivalent to tables and nested tables in Data Services. Segment fields are equivalent to columns. Each SAP message type has required segments and segment fields without which the IDoc will not be posted to SAP.
  The *Query Transform Definition* window in Data Services displays SAP segments (as tables) and SAP segment fields (as columns with data types). However, the *Query Transform Definition* window does not indicate which segments are required, nor does Data Services enforce SAP's segment repetition rules. An attribute of a segment is that it has a minimum and maximum number of times that it can occur in a single IDoc.

View SAP front-end applications to ensure you have the required order numbers and segments and your IDoc will not violate segment repetition rules. This information is also available in Data Services in the object's class attributes metadata.

Example 2

If your IDoc is a new sales order, then the sales order must:

- Be defined with valid a IDoc type, partner number, partner type, and message type names.
  Partner numbers and types are used to distribute IDoc data in SAP applications. If they do not exist or you do not specify the correct values, your IDoc will not post to SAP successfully. Use SAP front-end applications to configure SAP to receive an inbound IDoc (IDoc target) from Data Services.
  SAP applications define message types for different IDoc tasks. The message type you choose must be appropriate for the processing goal of the IDoc. Message types provide part of the required description of what is being passed to SAP. Again, reference the list of message types using SAP front-end applications.
- Match SAP segment field formatting.
- Contain all mandatory IDoc segments for its message type.

12.4.6.3.2.1.2 Data types and formats

Data Services supports several data types. IDoc Targets support only varchar.

- If an IDoc Target has columns configured in the varchar data type, no conversion occurs when it is sent to an SAP application.
- If an IDoc Target column uses another data type, the data type is converted to varchar and a default ABAP format.

The following table lists Data Services data types and their default formats:

Data Services data type	Is converted to ABAP...
date	date (YYYYMMDD)
datetime	date (YYYYMMDD)
decimal	float
double	float
int	int
interval	float
long	int
real	float
time	time (HHMISS)
varchar	no conversion

These default formats are defined by Data Services to help with the data movement to SAP applications. However, if the IDocs or the ABAP modules for IDoc processing in SAP are customized, you are responsible for modifying the default formats as needed to match the requirements.

- An IDoc can be customized in SAP
  Importing the metadata for your IDoc target can limit the risk of having a failed posting to SAP due to inconsistent formats.

- Input and posting modules can be customized
  IDocs are generated and read by ABAP modules. IDocs are processed in SAP applications first by an input module and then by a posting module. ABAP program modules control how data is interpreted. The configuration of these modules can be customized.
  For example, a posting module might be coded to receive a date formatted such as MMDDYYYY or DDMMYYYY instead of the standard ABAP date type format YYYYMMDD.

You must modify the metadata in Data Services or the format of the IDoc data generated in SAP to account for customized formats.

12.4.6.3.2.2 Adding an IDoc target to a data flow

When you drag an IDoc into a data flow, Data Services presents a pop-up menu of options for the type of data flow you are using.

To add an IDoc target to a data flow:

1. Select an IDoc from the object library.
2. Drag it into an open real-time or batch data flow.
3. Select *Make IDoc file target* or *Make IDoc message target*.
4. Add one or more source objects and a Query.

   You must copy imported IDoc file and message targets from the IDoc message target definition window into the target schema of a Query before using them in Data Services data flows.
5. Connect the source object(s) to the Query and the Query to the IDoc target object.
6. Click the name of the IDoc target.
7. Complete the IDoc target editor boxes. See *Target* [page 2548] for more information.
8. Click *OK*.

12.4.7 File formats

This section discusses SAP application file formats and how they relate to Data Services.

Related Information

Designer Guide: File Formats [page 254]

12.4.7.1 The Transport_Format

The Transport_Format is a Data Services predefined file format object for reading flat files from SAP applications. It is automatically used to define data transport objects added to SAP application data flows.

The Transport_Format appears in the *Format* tab of the object library under *Flat Files*.

You can modify it for your own use. You cannot delete it.

12.4.7.1.1 To edit the Transport_Format

1. From the object library, right-click Transport_Format and select *Edit*.

   The file format editor opens.
2. Notice that the *Type* is set to *SAP transport*, the *Name* is set to *Transport_Format*, and the *Delimiters*, which indicates where one column ends and another begins, uses the ASCII characters /127 as the column marker.

   The default delimiter for the Data Transport object is /127.

12.4.7.2 Defining an SAP applications file format

When you want to read from or write to a file in an SAP data flow and do not want to use the predefined Transport_Format, you can create your own SAP application file format.

1. Open the object library.
2. Click the *Formats* tab.
3. Right-click *Flat Files* and choose *New*.

   The file format editor opens.
4. Set the *Type* to *SAP transport*.
5. For the *Name*, use the automatically generated name, or enter a name to identify the format inside Data Services.

   The name can include alphanumeric characters and underscores (_). It cannot contain blank spaces.
6. (Optional) To use an existing file as a template for a file format, point to a file using the *Data File(s)* attribute.
   a) Under *Location*, select *Local* to indicate that you want to use a file on the computer where the Data Services Designer is running, or select *Job Server* to choose a file residing on the same computer as the default Job Server you associated with the repository.
   b) When you select *Job Server*, the browse options are disabled, so you must type the absolute path to the File name(s), or you can use a substitution parameter.

      An absolute path on UNIX might look like this:

      ```
      /user/data/abc.txt
      ```

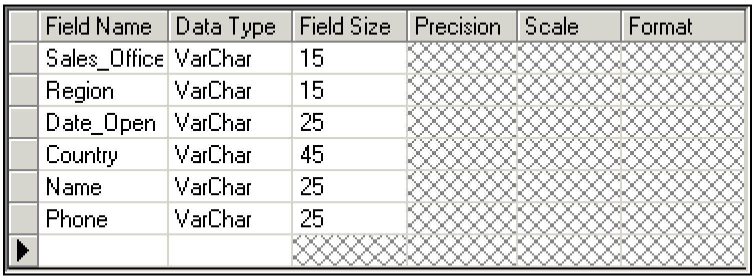
      A path on Windows might look like this:

      ```
      C:\DATA\abc.txt
      ```

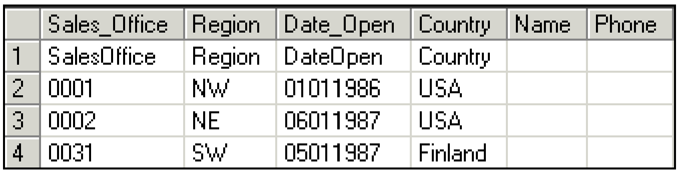
   > Note
   >
   > The Data File(s) attribute is a design-time tool that allows you to view data and create metadata schema using a data file located on the local or the Job Server computer. When the file format you create is dragged into a data flow as a source or target, you might need to change the File name and Root Directory

attributes to reflect the data files used in your job. Your data file(s) must be located relative to the Job Server computer on which your job will be executed.

When you specify the file, the editor populates the Column Attributes work area for editing and places sample data from the file in the Data View work area. As you edit column attributes, the Data View work area changes to reflect those modifications.



	Field Name	Data Type	Field Size	Precision	Scale	Format
	Sales_Office	VarChar	15			
	Region	VarChar	15			
	Date_Open	VarChar	25			
	Country	VarChar	45			
	Name	VarChar	25			
	Phone	VarChar	25			
▶						



	Sales_Office	Region	Date_Open	Country	Name	Phone
1	SalesOffice	Region	DateOpen	Country		
2	0001	NW	01011986	USA		
3	0002	NE	06011987	USA		
4	0031	SW	05011987	Finland		

Click any cell to edit it. Use mouse or keyboard navigation (tab, arrow keys) to move through the fields and view lists from which you can choose data. Right-click to view a pop-up menu to insert or delete new rows.

7. (Optional) Modify the file format delimiter.
   a) Under *Delimiters* click the *Column* box.
   b) Click the down arrow to view menu options.
   c) Click to choose a delimiter type (*Comma*, *Semicolon*, *Space*, or *Tab*). Alternately, enter an ASCII character. Valid ASCII characters range from /0 to /254.
   d) If defining a file format for a source, verify that the column delimiter matches the file from which you are reading. If defining a file format for a target, you can keep the default value.

   Note

   The data in the file cannot contain the character(s) used as delimiters.

8. (Optional) To examine the ATL file associated with this file format, click *Show ATL* to view a read-only version of the ATL file.

   To search for specific words or phrases within the file, click *Find* and enter a key word or phrase, then click *Find Next* to find occurrences of the word or phrase. Click *Cancel* when you are finished finding words or phrases. Click *Cancel* again to return to the file format editor.
9. After you edit your column attributes, click *Save & Close*.
10. If you are using an existing file as a model, verify that the fields are delimited and have the data type you expect.

Related Information

File format [page 2538]
Designer Guide: File format [page 254]

12.4.8 Real-time jobs

Data Services supports real-time data transformation including receiving messages from ERP systems (such as SAP ERP) or XML-based, analytic applications. “Real-time” means that Data Services reacts to messages as they are sent, performing predefined operations and responding appropriately.

You define the operations performed to process messages and create responses by building real-time jobs using the Data Services Designer and then by converting those jobs to on-demand services using the Data Services Administrator.

This section discusses real-time jobs in the SAP applications environment.

Related Information

Designer Guide: Real-time jobs [page 372]

12.4.8.1 SAP application table sources in real-time jobs

You can use SAP tables as sources in real-time jobs after you import metadata for the table into your repository. When you drag an SAP table to a data flow definition, it becomes a source.

When the data flow performs a query against the SAP table, it executes an SAP function call to extract the data through the SAP application server.

If you include an SAP table in a join with a real-time message source, Data Services includes the data set from the real-time message source as the outer loop of the join. If more than one supplementary source is included in the join, you can control which table is included in the next outermost loop of the join using the join ranks for the tables.

12.4.8.1.1 To use an SAP application table in a data flow

1. Import the metadata for the table using a datastore.

   The table appears in the table category under the datastore in the object library.
2. Drag the table into the data flow.
3. Connect a query to the output of the table.

Because SAP application tables are designed to extract a limited amount of data, the first step after the source would most likely be a query with a WHERE clause that limits the data extracted.

4. In the WHERE clause of the query, add an expression to reduce the amount of data extracted from SAP by this source.

   The join expression in the ON clause of the query includes specific data from the message that the data flow is processing. For example, if the data flow is using the ERP table VBUK to supply delivery status values to respond to an e-commerce application, the join expression would restrict the values from VBUK to the specific sales order referred to in the incoming message. In this case, the join expression restricts the values to sales order number and line item numbers matching the input.

12.4.8.2 Caching source data

All tables or files used as sources allow you to set a cache option. The option indicates that the data extracted is stored in memory until the data flow processing is complete.

In real-time jobs, you should not cache sources unless the data being cached is small and is unlikely to be updated in the life of the real-time process.

In batch jobs, caching can improve the performance of data flow processing by reducing the number of times a set of data is read from the database or file source. In real-time jobs, however, the improvement in performance provided by caching is minimized by the likelihood that the job reads only a small amount of data from the source for any given message. In addition, because the job reloads cached data only when the Access Server shuts it down and restarts it, cached data may become stale in memory.

12.4.8.3 Data cache targets and transactional loading

In addition to IDocs, you can use tables and files as targets in real-time jobs. Target tables in data flows also support transactional loading, in which the data resulting from the processing of a single message can be loaded into multiple tables as a single transaction. No part of the transaction is applied if any part fails.

You can specify the order that tables are included in the transaction through an option in each target table editor.

Loading targets as a single transaction only applies to targets in a single datastore. If the real-time job loads tables in more than one datastore, targets in each datastore are loaded independently.

In addition, the options to bulk load a table or to include pre-load or post-load commands are not available for tables used as targets in transactional loading.

Related Information

Reference Guide: Objects [page 832]

12.5 Connecting to SAP NetWeaver Business Warehouse

To use an SAP data source as a Data Services source, you must define a Data Services datastore to serve as the logical link to SAP, then import your metadata into the Data Services datastore.

Related Information

Designer Guide: Datastores [page 209]

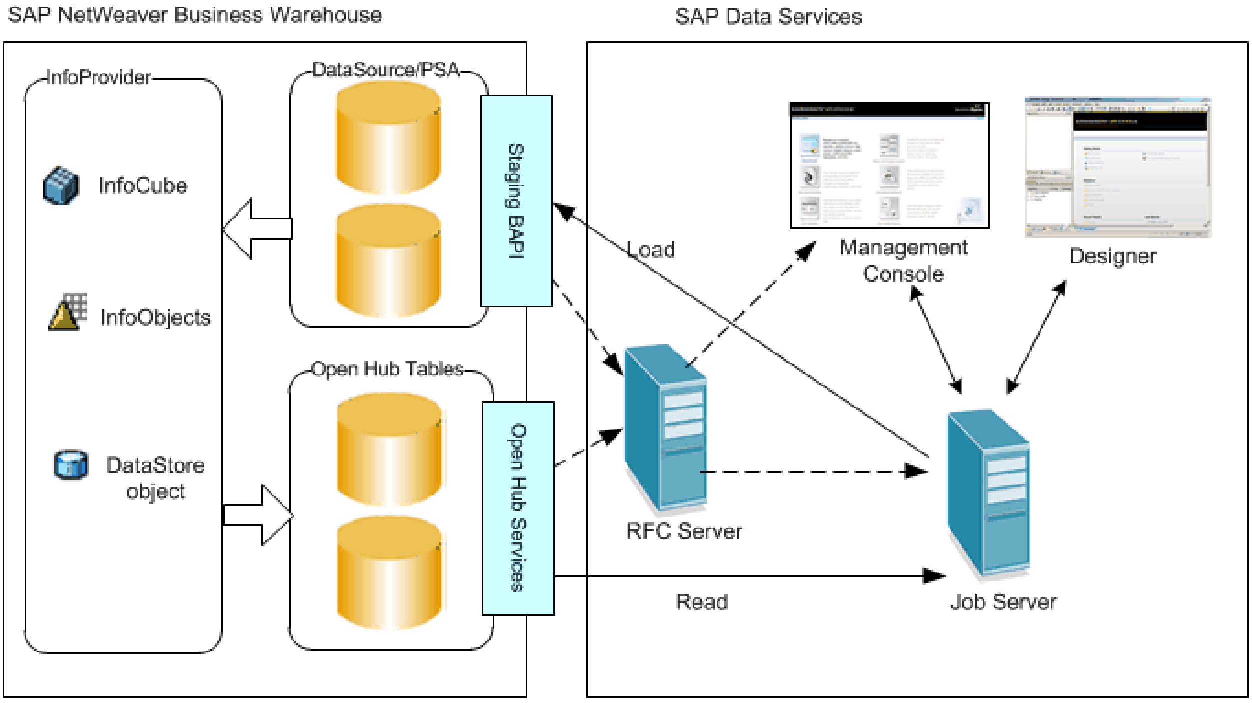
12.5.1 Using Data Services in SAP NetWeaver Business Warehouse environments

SAP Data Services provides the following capabilities in an SAP NetWeaver Business Warehouse (BW) environment:

- Read data from an InfoProvider (such as an InfoCube, DataStore object, and InfoObject) in one of the following ways:
    - SAP Open Hub Destination service which executes an SAP NetWeaver BW process chain that reads the data from the InfoProvider and loads an Open Hub Destination table that Data Services reads as a source table.
    - Auto-generated ABAP program or RFC functions that read data from the underlying tables of the InfoCube, DataStore object, or InfoObject.

  The preferred method is to use the SAP Open Hub Destination service to read from a BW InfoProvider.
- Load data into an SAP NetWeaver BW InfoSource by using a Data Services batch job which you can start in one of the following ways:
    - Execute a Data Services batch job that calls a BW InfoPackage automatically to load a BW InfoProvider.
    - Use an InfoPackage in SAP NetWeaver BW to schedule and execute the Data Services load job which has been exported to SAP NetWeaver BW.

The following diagram shows a high-level overview of the components involved with reading data from and loading data to an SAP NetWeaver BW system.



This diagram shows that:

1. Data Services can load data into a DataSource in the Persistent Storage Area (PSA) using the Staging Business Application Programming Interface (BAPI), with support from the RFC Server.
2. Data Services can read data from a BW system by using the Open Hub Service with support from the RFC Server.

For more information about Data Services and BW, see the following topics::

- *Netweaver 7.3: Transferring Data with SAP Data Services*
- *Netweaver 7.4: Transferring Data with SAP Data Services*

Related Information

Reading from SAP NetWeaver BW [page 2515]
Loading into SAP NetWeaver BW [page 2524]

12.5.2 SAP NetWeaver BW datastores

SAP NetWeaver Business Warehouse (SAP NetWeaver BW) is a business warehouse application that runs on an SAP application server.

You can extract data from the SAP source, transform it as specified in a Data Services batch job, and load it into various targets including a predefined SAP BW InfoSource.

Data Services provides two datastore types for SAP BW: one for sources and one for targets. For other datastore types, you create a datastore, import objects such as tables, then create sources and targets using those objects. For SAP BW, you create separate source and target datastores because they expose different objects.

12.5.2.1 SAP NetWeaver BW as a source

SAP NetWeaver BW can be used as a source in Data Services by using a SAP BW Source datastore. You can import the same objects as in SAP Applications datastores except hierarchies. In addition, you can browse and import InfoAreas, InfoCubes, ODS objects, and Open Hub tables.

To use SAP NetWeaver BW data as a source:

- Define an SAP BW Source datastore that will serve as the logical link to your BW system.
- Import the SAP BW metadata into Data Services using the SAP BW Source datastore.

12.5.2.1.1 Defining an SAP BW Source datastore

To define an SAP BW Source datastore:

1. In the Designer's object library, select the *Datastores* tab.
2. Right-click in a blank area of the object library window and select *New*.
   The *Create New Datastore* window opens.
3. Enter a name for the datastore in the *Datastore name* box.

   The name can contain any alphabetic or numeric characters or underscores (_). It cannot contain spaces.
4. For the *Datastore type* box, select *SAP BW Source*.
5. In the *Application server* box, type the host name.

   The host name can be either the host name or an SAP routing string plus the host name. An example of valid syntax for an SAP routing string is:

```
/H/<IP_Address_of_local_SAP_router>/H/<IP_Address_of_target_SAP_router>/H/
<IP_Address_of_target_application_server>
```

6. Type the *User name* and *Password*.
7. Click *Advanced* to enter properties for the SAP application server on which BW is running.

   The recommended method to extract data from a BW system is through the Open Hub Destination service. However, it is still possible to use an ABAP program or RFC functions to extract data from the underlying tables of the InfoAreas, InfoCubes, and DataStore objects.

   a) The following Advanced options are always required regardless of which method you use to extract data from a BW system. Ensure that the default values for these Advanced options are what you want:

      ○ *Language*
      ○ *Code page*
      ○ *Client number*
      ○ *System number*

   b) If you want to use an ABAP program or RFC functions, you must also specify *Data transfer method* and the associated options for each transfer method (for details, see the link in Related Topics below).

8. Click *OK* to add the SAP BW Source datastore.

   The object library displays the newly created SAP BW Source datastore.

You can now browse and import the Open Hub tables or the BW InfoProvider tables that will serve as your SAP BW source.

Related Information

Datastore [page 2531]

12.5.2.1.2 Using the sapnwrfc.ini file

To change or add parameters without modifying the datastore settings, you can specify them in an `sapnwrfc.ini` file.

Place your sapnwrfc.ini file in the same directory as the RFC client/server program (for Data Services, commonly `%LINK_DIR\bin`). Or you can specify the DS_NWRFC_INI environment variable in the operating system. For example, you could can place the sapnwrfc.ini file in `C:\Temp` and specify `DS_NWRFC_INI=C:\Temp`.

You can specify an `sapnwrfc.ini` file for two different applications in Data Services:

- To use an sapnwrfc.ini file to override datastore settings, edit the SAP datastore in the Designer. From the object library, open the editor for the SAP datastore to configure and click *Advanced*. Under the *SAP* option category, for *Use sapnwrfc.ini*, select *Yes* and enter the appropriate *Destination name*.
- To use an sapnwrfc.ini file for real-time IDocs, configure the RFC client in the Administrator. Expand the *Real-Time* node, and for the Access Server to configure, click *Client Interfaces*. Open an instance of the RFC client, and select the check box to *Use sapnwrfc.ini* and enter a destination name.

> i Note
>
> Data Services ignores the parameters USER, PASSWD, and CLIENT in sapnwrfc.ini because storing these security-related parameters could introduce a security risk.

For more information about how to use the `sapnwrfc.ini` file, refer to the SAP documentation at *http://help.sap.com/saphelp_nwpi711/helpdata/en/48/ce50e418d3424be10000000a421937/frameset.htm*.

12.5.2.1.3 Browsing and importing metadata for an SAP BW source

You can import metadata from the object library in several ways:

- By name when you know the name of the object you want to import
- By browsing when you do not know the name of the objects

- By searching when you know the partial name of the object and you have a large number of objects

12.5.2.1.3.1 To import SAP NetWeaver BW source metadata by name

1. In the object library, select the name of your SAP NetWeaver BW Source datastore.
2. Right-click and choose *Import By Name*.
3. In the *Import by Name* window, select the type of metadata that you want to import from the Type drop-down list.
    - If you select *Open Hub Table*, type the name of the table in the *Name* field.
    - If you select InfoArea, InfoCube or ODS object, the *Name* field offers a list of available objects. Select the name of the object you want to import.
4. Click *OK*.

    The specified source appears in the object library.

12.5.2.1.3.2 To import SAP NetWeaver BW source metadata by browsing

1. In the object library, right-click the name of the BW datastore.
2. Select *Open*.
    The Repository Metadata opens in the workspace.
3. Select *External Metadata* to view the BW objects that you can import.

    The BW external metadata browser opens with the Open Hub Tables node expanded by default. You can scroll down to an InfoArea, InfoCube, or DataStore object, expand the object, and browse its lower level content to identify the objects that provide data to it.
4. Select the structures to import.

    For Open Hub Tables, you can select:
    - A parent category (imports all Open Hub Tables)
    - A single Open Hub Table
    - A subset of Open Hub Tables by using the CTRL or CTRL + SHIFT keys

    For InfoAreas, InfoCubes, or DSOs, you can select:
    - A parent category (imports the parent and all its children)
    - A single category
    - A subset of objects by using the CTRL or CTRL + SHIFT keys
5. Right-click and choose *Import*.
    The imported structures appear in the object library. You can also open the datastore to see the contents of the datastore in the workspace by using the Repository Metadata option.

12.5.2.1.3.3 To import SAP NetWeaver BW source metadata by searching

1. In the object library, select the name of your SAP NetWeaver BW Source datastore.
2. Right-click and choose *Search*.
3. Use the Search dialog to find and import metadata.

12.5.2.2 SAP NetWeaver BW as a target

Data Services can use SAP NetWeaver BW as a target by using an SAP BW Target datastore.

You can execute the job directly from the Data Services Designer or from SAP NetWeaver BW using BAPI functions through the Data Services RFC Server.

12.5.2.2.1 SAP NetWeaver BW loading profile

If you are using Data Services to load data into SAP NetWeaver BW during production, create an SAP application profile, such as BODS_BW_PROD, that defines additional authorizations for PRODUSER. This profile requires the following authorizations:

- S_RS_ADMWB
- S_RFC

Related Information

S_RS_ADMWB [page 2428]
S_RFC [page 2427]

12.5.2.2.2 To use an SAP NetWeaver BW InfoSource as a Data Services target

1. Define a Data Services SAP BW Target datastore that will serve as the logical link to your SAP system.
2. Import your SAP NetWeaver BW InfoSource metadata into the Data Services SAP BW datastore.

Related Information

Defining an SAP BW Target datastore [page 2513]

Browsing and importing metadata for an SAP BW target [page 2514]

12.5.2.2.3 Defining an SAP BW Target datastore

To define an SAP BW Target datastore:

1. In the Designer's object library, select the *Datastores* tab.
2. Right-click in a blank area of the object library window and select *New*.

   The *Datastore Editor* window opens.
3. Enter a name for the datastore in the *Datastore name* box.

   The name can contain alphanumeric characters or underscores (_). It cannot contain spaces.
4. For the *Datastore type* box, select *SAP BW Target*.
5. In the *Application server* box, type the SAP host name.
6. Type the *User name* and *Password*.
7. Click *Advanced* to verify the following required properties for the SAP application server on which BW is running:
   - *Language*
   - *Code page*
   - *Client number*
   - *System number*
   - *Routing string*

   Include routing string information if your application server is a remote system accessed through a gateway. No SAP application routing permissions are required, but you must use the specified syntax in the *Routing string* field to ensure connection. An example of valid syntax for an SAP routing string is:

   ```
   /H/<IP_Address_of_local_SAP_router>/H/<IP_Address_of_target_SAP_router>/H/
   <IP_Address_of_target_application_server>
   ```
8. Click *OK* to add the datastore.
   SAP BW Target datastores display in the object library.

You can now browse and import the metadata for the Master and Transaction Transfer Structures that will serve as your SAP BW target.

For more information, see “Internationalization” in the SAP Library.

Related Information

Datastore [page 2531]
SAP Library - Internationalization

12.5.2.2.4 Using the sapnwrfc.ini file

To change or add parameters without modifying the datastore settings, you can specify them in an `sapnwrfc.ini` file.

Place your sapnwrfc.ini file in the same directory as the RFC client/server program (for Data Services, commonly `%LINK_DIR\bin`). Or you can specify the DS_NWRFC_INI environment variable in the operating system. For example, you could can place the sapnwrfc.ini file in `C:\Temp` and specify `DS_NWRFC_INI=C:\Temp`.

You can specify an `sapnwrfc.ini` file for two different applications in Data Services:

- To use an sapnwrfc.ini file to override datastore settings, edit the SAP datastore in the Designer. From the object library, open the editor for the SAP datastore to configure and click *Advanced*. Under the *SAP* option category, for *Use sapnwrfc.ini*, select *Yes* and enter the appropriate *Destination name*.
- To use an sapnwrfc.ini file for real-time IDocs, configure the RFC client in the Administrator. Expand the *Real-Time* node, and for the Access Server to configure, click *Client Interfaces*. Open an instance of the RFC client, and select the check box to *Use sapnwrfc.ini* and enter a destination name.

> Note
>
> Data Services ignores the parameters USER, PASSWD, and CLIENT in sapnwrfc.ini because storing these security-related parameters could introduce a security risk.

For more information about how to use the `sapnwrfc.ini` file, refer to the SAP documentation at *http://help.sap.com/saphelp_nwpi711/helpdata/en/48/ce50e418d3424be10000000a421937/frameset.htm*.

12.5.2.2.5 Browsing and importing metadata for an SAP BW target

To access InfoSource metadata in Data Services, import its metadata into the object library.

You can import InfoSource metadata in one of two ways:

- By name
- By browsing

12.5.2.2.5.1 To import InfoSource metadata by name

1. In the object library, select the name of the SAP BW Target datastore.
2. Right-click, and choose *Import By Name*.
3. In the Import by Name window, specify the Source System and InfoSource names.

   Also, specify whether the InfoSource type is Master or Transaction. If the InfoSource type is Master, specify whether the subtype is Text, Attributes, or Hierarchy.
4. Click *OK*.

   The imported objects appear in the object library under the target datastore node.

12.5.2.2.5.2 To import InfoSource metadata by browsing

1. In the object library, right-click an SAP BW datastore.
2. Choose *Open*.

   The metadata browser opens in the workspace.

   From this viewer you can browse both Master and Transaction InfoSources. Both are sorted in alphabetical order.

   Master Transfer Structures contain dimension data, and Transaction Transfer Structures contain fact data.

   A single InfoSource can be loaded from multiple Source Systems. If you expand an InfoSource you can see all the Source Systems that provide data to it.
3. Select the structures to import.
4. Right-click and choose *Import*.

   The imported structures appear in the object library.

12.5.3 Reading from SAP NetWeaver BW

SAP NetWeaver Business Warehouse provides an Open Hub Service through which SAP Data Services can read data from an InfoProvider. The Open Hub Service extracts data from an InfoProvider and loads it into an Open Hub Destination object that Data Services reads. Data Services uses the Destination Type Third Party Tool.

The following is an overview of the tasks that you must do to read from an Open Hub Destination table:

- In the SAP Data Warehousing Workbench, define an RFC destination and set up the Open Hub Destination object.
- In the Data Services Management Console, set up the RFC server in the Administrator.
- In the Data Services Designer, do the following:
  - Define an SAP BW Source datastore to connect to the BW System that has the Open Hub Destination table.
  - Import the Open Hub Destination table.
  - Create an Data Services job to start the BW process chain that loads data from the InfoProviders into an Open Hub Destination table, receive notification when the data is available, read the data, and delete the data from the Open Hub Destination table after successfully reading the data.

Related Information

Setting up the Open Hub Destination object [page 2516]
Setting up the RFC server to read an Open Hub table [page 2516]
Defining an SAP BW Source datastore [page 2509]
Importing Open Hub tables [page 2517]
Creating a job to read an Open Hub table [page 2518]
Running the job to read an Open Hub table [page 2518]

12.5.3.1 Setting up the RFC server to read an Open Hub table

The Remote Function Call (RFC) Server is required when a job reads from an Open Hub Destination table in SAP NetWeaver Business Warehouse (BW). When you execute the SAP Data Services job, Data Services first launches the BW process chain to load the Open Hub Table. The RFC Server notifies Data Services when the load has completed. The Data Services job can then read data from the Open Hub Table, transform it, and load the data to a target.

To set up the RFC Server to read Open Hub Destination tables, you must do the following tasks:

- In the SAP Data Warehousing Workbench, define an RFC Destination:
  - Specify a name in *RFC Destination*.
  - Ensure that *Connection Type* has the value **`TCP/IP Connection`**.
  - On the *Technical Settings* tab, in the *Registered Server Program* area, specify a name for your RFC Server in *Program ID*.
    The value in *Program ID* is the value that you specify in option *RFC Program ID* when you configure the RFC Server in Data Services.
- In the Management Console of Data Services, go to the Administrator and do the following to configure the RFC Server:
  - Select ▶ *Management* ▶ *Repositories* ◢ on the navigation tree, and add the repository where you imported the Open Hub table.
  - Select ▶ *SAP Connections* ▶ *RFC Server Interface* ◢ on the navigation tree, and configure the RFC Server associated with the Open Hub table.
    In the option *RFC Program ID*, specify the value of option *Program ID* when you defined the RFC Server in the Data Warehousing Workbench. The value of attribute *3RDPARTYSYSTEM* for the Open Hub table is also the value of *Program ID*.
  - Start the RFC Server.

> i Note
>
> You cannot have multiple RFC server instances that refer to the same Program ID running in the Data Services environment.

Related Information

Management Console Guide: Administrator, Adding an RFC server interface [page 1935]

12.5.3.2 Setting up the Open Hub Destination object

Use the SAP Data Warehousing Workbench to define an RFC destination and set up an Open Hub Destination object.

1. Define an RFC Destination that points to Data Services RFC Server registered Program ID.

- ○ Specify a name in *RFC Destination*.
- ○ Ensure that *Connection Type* has the value **`TCP/IP Connection`**.
- ○ On the *Technical Settings* tab, in the *Registered Server Program* area, specify a name for your RFC Server in *Program ID*.
  The value in *Program ID* is the value that you specify in the option *RFC Program ID* when you configure the RFC Server in the Data Services Administrator on the Management Console.

2. Create an Open Hub Destination and specify the following options:
   - ○ For *Destination Type*, select *Third Party Tool* in the drop-down list.
   - ○ For *RFC Destination*, specify the name that you defined in step 1.
3. Create a Data Transfer Process (DTP) to extract data from an InfoProvider and load it to the Open Hub Destination table.
4. Create a process chain to execute the DTP.

   i Note

   Each process chain can only contain one DTP of the Open Hub Destination.

12.5.3.3 Importing Open Hub tables

The Open Hub Destination tables must already be defined in SAP NetWeaver BW before you can import their metadata into SAP Data Services. When you browse the list of Open Hub tables in the SAP BW Source datastore in Data Services, only the Open Hub objects with destination type Third Party Notification are visible.

i Note

Although import by browsing might be the simpler way to import Open Hub tables, the following procedure describes how to import by searching to provide an example if you have a BW system with a large number of objects and browsing might be more difficult.

1. In the object library, select the name of your SAP NetWeaver BW Source datastore.
2. Right-click and choose *Search*.
3. In the *Search* window, do the following actions to find and import metadata.
   a) Ensure that the value for *Select location* is *External Data*.
   b) In the *Object Type* drop-down list, select *Open Hub Tables*.
   c) Choose the *Select criteria* you want and type in the string you want to find.
   d) Click *Search*.
   e) From the search result list, select the table names that you want, right-click, and choose *Import*.

The *Local Object Library* shows the imported objects under the Open Hub Tables node under the datastore name.

Related Information

To import SAP NetWeaver BW source metadata by name [page 2511]

To import SAP NetWeaver BW source metadata by browsing [page 2511]
To import SAP NetWeaver BW source metadata by searching [page 2512]

12.5.3.4 Creating a job to read an Open Hub table

This topic describes the usage scenario of one user reading from one Open Hub table.

> Note
>
> When you import Open Hub Tables, you can use them in regular data flows. If you import InfoAreas, InfoCubes, or DataStore objects, Data Services imports the underlying ABAP tables which you use in ABAP data flows so that you can push down the ABAP functions. If you use the ABAP tables in regular data flows, you cannot push down the ABAP functions.

1. In the SAP Data Services Designer, create a job, a work flow, and data flow within the job.
2. Select an Open Hub table from the local object library and drag it into the open data flow.
   The Open Hub table becomes a source object in the data flow.
3. Double-click the Open Hub source object in the data flow to open the editor and specify the following options.
   a) For *Open Hub Table*, leave the default value (name of the current Open Hub table).
   b) Select *Execute process chain before reading* to start the SAP NetWeaver BW process chain that loads the Open Hub table from which the Data Services Open Hub source object reads the data.
   c) For *Process chain name*, ensure that the default value (first process chain name listed in the attributes for the current Open Hub table) is the correct process chain.

   For details about the options on the Open Hub Table editor, see the Related Topic below.
4. Add the transforms you want to perform on the extracted data and add your target to the data flow.
5. If you want to take specific actions for any errors that might occur while accessing and reading the Open Hub table, add a try/catch block and select the exception group SAP BW execution errors (1009).
6. Save the job and data flow.

Before you execute the job, you must set up the RFC server.

Related Information

Setting up the RFC server to read an Open Hub table [page 2516]
Open Hub Table source [page 2565]
Designer Guide: Work Flows, Defining a try/catch block [page 338]

12.5.3.5 Running the job to read an Open Hub table

The SAP Data Services job initiates the process chain to load the Open Hub Destination table in SAP NetWeaver BW.

The following must be running for job execution to complete successfully:

- The SAP NetWeaver BW application server where the Open Hub Destination tables reside
- The SAP Data Services RFC Server
- The SAP Data Services Job Server

1. From the Designer of the SAP Data Services, open the project that contains your Open Hub job.
2. Right-click the job name and select *Execute*.
   The *Trace log* window opens and you can see the job progress messages. A message appears when the process chain completes.

   If an error occurs, the Error log displays the status of the process chain execution. For information about the status codes for the process chain execution, see the Related Topic below.

Related Information

sap_openhub_processchain_execute [page 1654]

12.5.3.6 Reading multiple Open Hub tables

This section describes examples of when you might want to read from multiple Open Hub tables.

12.5.3.6.1 Creating a data flow for multiple users to read one InfoProvider

In this scenario, two users want to extract data from the same InfoProvider (InfoCube or DataStore object) in SAP NetWeaver BW to transform and load to a target in SAP Data Services. For example, one user might want to obtain information about materials in Europe and a second user might want the same materials information in North America. Both users can use the same Data Services data flow and specify their specific Open Hub table and process chain with substitution parameters.

1. In the BW Administrative Workbench, define two different Open Hub Destinations that have the same schema. For each Open Hub Destination, define a Data Transfer Process (DTP) that extracts the data with a filter for each geographic area.
   For example, you might define `OpenHubEurope` and `OpenHubNA`.
2. In the Designer of Data Services, in your Open Hub datastore in the *Local Object Library*, import the Open Hub Destination tables.
3. Define substitution parameters for the Open Hub table name and process chain name. To add and define a substitution parameter:

   a) In the Designer of Data Services, select *Tools* > *Substitution Parameter Configurations*.
   b) In the *Substitution Parameter Editor*, scroll to the bottom of the parameter list and double-click the blank cell to enable editing.

   c) Type the name of your substitution parameter, prefaced with **`$$`**.
   For example, you can enter **`$$OpenHubTable`**.
4. You can define default values in a substitution parameter configuration and associate these defaults with a system configuration.
   For example, you might have an existing system configuration for Europe and one for North America.
   a) Create a substitution parameter configuration for each system configuration: **`Sub_Parm_Config_Europe`** and **`Sub_Parm_Config_NA`**.

      For details about creating a substitution parameter configuration and associating it with a system configuration, see the Related Topic below.
   b) To define default values, double-click in the blank cell next to the substitution parameter name in the column that identifies each substitution parameter configuration and type the constant value.
      For example, you might type **`OpenHubEurope`** as the default value for the substitution parameter *$$OpenHubTable* in the column for *Sub_Parm_Config_Europe*. Similarly, you might type **`OpenHubNA`** in the column for *Sub_Parm_Config_NA*.
5. Create a Data Services data flow.
6. Select an Open Hub table from the *Local Object Library* and drag it into the open data flow.
   The Open Hub table becomes a source object in the data flow.
7. Double-click the Open Hub source object in the data flow to open the editor and specify the following options.
   a) For *Open Hub Table*, click the arrow to display the drop-down list, and select the substitution parameter that you defined for the name of the Open Hub Table in step 2.
   b) Select *Execute process chain before reading* to start the SAP NetWeaver BW process chain that loads the Open Hub table from which the Data Services Open Hub source object reads the data.
   c) For *Process chain name*, click the arrow to display the drop-down list, and select the substitution parameter that you defined for the process chain name in step 2.

   For details about the options on the Open Hub Table editor, see the Related Topic below.
8. Add the transforms you want to perform on the extracted data and add your target to the data flow.
9. If you want to take specific actions for any errors that might occur while accessing and reading the Open Hub table, add a try/catch block and select the exception group SAP BW execution errors (1009). For details about how to define a try/catch block, see the Related Topic below.
10. Save the data flow.
11. Before you execute the job, ensure that the RFC server has been set up on the Management Console Administrator:
    - Configure the repository where you imported the Open Hub table.
    - Define the RFC Server associated with the Open Hub table.
    - Start the RFC Server.
12. When you execute the job, select the system configuration associated with your substitution parameter configuration.
    - In the Management Console Administrator, you can select the system configuration on the following pages:
      - Execute batch job.
      - Schedule batch job.
      - Export execution command.

      To override the default value of a substitution parameter, use the *Add Overridden Parameter* option on these pages.

- In the Data Services Designer, you can select the system configuration on the *Execution Options* tab of the *Execution Properties* window. To override the default value of a substitution parameter, use the *Pick a Substitution Parameter* option on the *Substitution Parameter* tab.

Related Information

Designer Guide: Variables and Parameters, Associating a substitution parameter with a system configuration [page 418]

Open Hub Table source [page 2565]

Designer Guide: Work Flows, Defining a try/catch block [page 338]

Setting up the RFC server to read an Open Hub table [page 2516]

12.5.3.6.2 Creating a data flow to join multiple Open Hub tables

In this scenario, one user reads two Open Hub tables to join the data from them. The Data Services job will consist of a work flow that contains the following:

- A script that contains an SAP function to launch the process chains that extract data from the InfoProvider objects and load it to tables in the Open Hub Destination.
- A data flow to read the Open Hub tables.
- A script to send the Open Hub read status to the BW system.

1. In the BW Administrative Workbench, define Process Chains, each with one Data Transfer Process that extracts the data from the InfoProvider and loads to the Open Hub Destination table.
2. In the Data Services Designer, import the Open Hub Destination tables that you want to join.
3. In the Data Services Designer, create a job, create a work flow, and add a script to the work flow.
4. Open the script and add the following:
   a) For each Open Hub table, add the function sap_openhub_processchain_execute to start the Process chain.
   The following sample function specifies the open hub table **`Materials`** and the process chain **`Materials_PC`**.

   ```
   sap_openhub_processchain_execute('open_hub_datastore','Materials',
   'Materials_PC',<$pclogid>,<$returntxt>)
   ```

   For details about this function, see the Related Topic.
   b) If you want to take specific actions for any errors that might occur while reading the Open Hub table, check the return value of the function sap_openhub_processchain_execute.
   For example, the following script commands check the return value, generate an exception, and print the error if the function is not successful.

   ```
   $status = sap_openhub_processchain_execute('open_hub_datastore',
   'Materials','Materials_PC',<$lpcogid>,
   <$returntxt>);
   If ($status != 'S') raise_exception ('Materials_PC process chain execution
   failed ' || $returntxt);
   ```

   c) If you want to take further actions for any errors that might occur while reading the Open Hub table, add a try/catch block around the data flow and select the exception group SAP BW execution errors (1009).

5. Create a data flow and, from the Local Object Library, drag in each Open Hub table that you want to join.
6. For this scenario, you do not need to open the Open Hub Table editor because you can leave the defaults for the following Open Hub source options:
   a) For *Open Hub Table*, leave the default value (the name of the current Open Hub table).
   b) *Execute process chain before reading* is unselected by default. Keep this default because the first script starts the SAP NetWeaver BW process chain that loads the Open Hub table from which the Data Services Open Hub source object reads the data.
   c) *Process chain name* will be disabled because *Execute process chain before reading* is not selected.

   For details about the options on the Open Hub Table editor, see the Related Topic below.
7. Add the Query transform to join the Open Hub tables, add any other transforms you want to perform on the extracted data, and add your target to the data flow.
8. Add a second script after the data flow in the work flow.
9. Open the second script and for each Open Hub table, add the function sap_openhub_set_read_status to the RFC Server to send the status of the Open Hub table read to the BW system.
   For example, the following function specifies the open hub destination **`Materials`**.

   ```
   sap_openhub_set_read_status('open_hub_datastore','Materials',
   <'X'>,<$returntxt>)
   ```

   The function sap_openhub_set_read_status deletes the data from the BW Open Hub Destination.
10. Save the job and data flow.
11. Before you execute the job, make sure you do the following:
    - Configure the repository where you imported the Open Hub table
    - Define the RFC Server associated with the Open Hub table
    - Start the RFC Server

Related Information

sap_openhub_processchain_execute [page 1654]

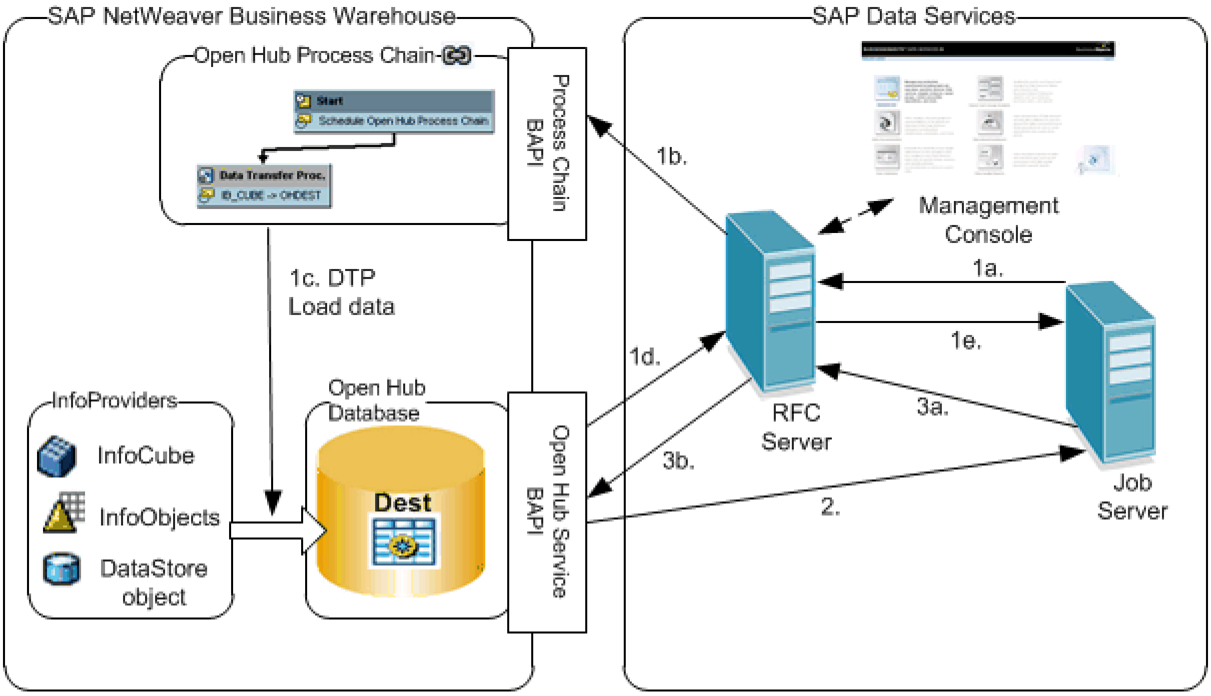
sap_openhub_set_read_status [page 1657]

Designer Guide: Work Flows, Defining a try/catch block [page 338]

Setting up the RFC server to read an Open Hub table [page 2516]

12.5.3.6.3 Data Services Open Hub Reader run-time architecture

The following diagram shows a high-level overview of the components involved when using the SAP Open Hub Destination service to read data from an SAP NetWeaver BW system.



This diagram shows:

1. The Data Services job executes function sap_openhub_processchain_execute which starts the following process:

   a.

   a. Sends a web service request to the Open Hub RFC Server to start the process chain.
   b. The RFC Server starts the Process Chain in BW system.
   c. The Process Chain contains a Data Transfer Process (DTP) that extracts data from the InfoProvider and loads the Open Hub Destination table.
   d. When the DTP completes, the Open Hub Service notifies the RFC Server.
   e. The RFC Server notifies the Job Server that the Open Hub table is loaded and returns the read request ID of the Open Hub Destination.
2. The Data Services data flow reads the data from the Open Hub table using read request ID.
3. The Data Services job contains a second script that executes the function `sap_openhub_set_read_status` which starts the following process:
   a. Sends a web service request to the Open Hub RFC Server to set the read status.
   b. RFC Server sets the read status in the Open Hub BW system which deletes the data from the Open Hub Destination table.

12.5.3.7 Open Hub table restrictions

The following restrictions exist when using Open Hub tables:

- Only one job at a time can read an Open Hub Destination table.
- A process chain of an Open Hub Destination can contain only one of its Data Transfer Processes (DTPs).
- Each process chain cannot contain DTPs for more than one Open Hub Destinations.

12.5.4 Loading into SAP NetWeaver BW

This section describes the following tasks:

- Preparing the job
- Setting up SAP NetWeaver BW InfoCubes and InfoSources
- Designating Data Services as a source system
- Job execution

You can execute Data Services jobs to load to SAP NetWeaver BW from the Designer or you can schedule them from SAP or Data Services.

> i Note
>
> This section assumes you are an experienced SAP NetWeaver BW user and have already followed similar procedures. Your system might be configured differently than in the examples, so treat the steps in the procedures as high-level information. You might need to provide additional details or do the steps in a somewhat different order.

12.5.4.1 Preparing the job

This topic provides an overview of how to prepare to load data into SAP NetWeaver BW.

First, in SAP NetWeaver BW define the appropriate SAP security profiles and authorizations.

In the Data Services Designer:

1. Add datastores for:
   - The system where your source data resides.
   - The system where your SAP NetWeaver BW InfoSource (Transfer Structure) resides (using an SAP BW Target datastore).
2. Import the source and target metadata.
3. Construct and validate the job that will extract data from the source to the SAP NetWeaver BW InfoSource.

In the Data Services Administrator:

1. Add the repository that includes the job.
2. Start the RFC server. In the Administrator, add an RFC Server Interface and start it from the status page.

Related Information

SAP NetWeaver BW as a target [page 2512]
Management Console Guide: Administrator, Starting or stopping an RFC server interface connection [page 1936]
Starting the RFC server [page 2486]
Executing the job [page 2527]

12.5.4.2 Setting up SAP NetWeaver BW InfoCubes and InfoSources

Use the SAP Data Warehousing Workbench to set up SAP BW.

To set up the environment:

1. Create and activate the InfoCube or InfoCubes where the extracted data will ultimately be placed.

   Your InfoCubes should be defined to hold data extracted using Data Services.
2. Create and activate the InfoSource where Data Services will place the extracted data:
   a) Go to the InfoSource window (*InfoSources* tab).
   b) Right-click InfoSources at the top of the hierarchy and select *Create Application Component*.
   c) Complete the window that appears with appropriate information. So for *Application comp*, you might enter: MYAPPCOMP. For *Long description*, you might enter: My application component.
   d) Click *Enter*.

      The application component is created and appears in the hierarchy list.
   e) Right-click the name of your new application component in the component list and select *Create InfoSource*.

      The Create InfoSource: Select Type window appears.
   f) Select the type of InfoSource you are creating (*Transaction data* or *Master data*) and click *Enter*.

      When you select Transaction data, the Create InfoSource (Transaction data) window appears.
   g) Enter the appropriate information and click *Enter*.

      The new InfoSource appears in the hierarchy under the application component name.

12.5.4.3 Designating Data Services as a source system

If the source system does not exist, right-click and choose *Create source system*, then create the new source system as explained in the following steps:

1. From the SAP Data Warehousing Workbench window, right-click *Source systems* at the top of the hierarchy, and choose *Create source system*.
   The *Select Source System Type* window appears.
2. Select the last check box on the window and click *Enter*.
   The *Create Source System* window appears.
3. Enter information about your system, then click *Enter*.
   The *RFC Destination* window appears.
4. Click *Registered Server Program*.
5. Enter the program ID.
6. Click the *Save* icon to complete registration.
7. Exit this menu and return to the Data Warehousing Workbench InfoSource window.
8. If the Source System already exists, right-click the InfoSource and select *Assign Source System*.
   The *Transaction data - InfoSource: Assign source system* window appears.
9. Enter the information required and click *Enter*.

10. Click the button to the right of the InfoSource name to view the Maintain Communication Structure window.
11. Add or change information as required.
12. Activate the Transfer Structure for this Source System:
    a) Right-click the InfoSource name and select *Update InfoSource Metadata*.
    b) Enter or change information as required, then click *Save*.

> Note
>
> If you will be performing both full and delta updates, check Delta update.

13. Map the transfer structure fields to the communication structure fields.

> Note
>
> You must successfully complete all previous steps to begin this step.

    a) Right-click the name of the external source system and select *Maintain Transfer Rules*.
    b) Map the fields and add transformation information as required.
    c) Click *Activate*.

12.5.4.4 Job execution

You can execute the job from the Designer or from SAP NetWeaver BW.

You can schedule the job to execute from SAP NetWeaver BW, which initiates the request that causes Data Services to extract data from the specified sources and make it available to SAP.

The batch job execution process consists of the following steps:

- If you are using the SAP NetWeaver BW scheduler, it executes code that sends the load request to the Data Services RFC server.
  The load request contains the following information:
  - The name of the batch file you exported from Data Services
  - The request ID, request date, request time, and full/delta update flag
- If you are using the SAP NetWeaver BW scheduler, the Data Services RFC server calls the Job Server which then invokes Data Services to extract and load the SAP NetWeaver BW InfoSource.
- Request information is passed to Data Services which executes the job and loads data to SAP NetWeaver BW with the request ID to identify into which transfer structure it is loading.
- Data Services extracts the data.
- Before loading to SAP NetWeaver BW (through the staging BAPI interface), Data Services first checks whether the InfoSource metadata changed since the last data import.
  If the metadata changed, Data Services produces an error message indicating that you must re-import the Transfer Structure metadata into the Data Services repository before proceeding to import the target data into the Transfer Structure.

12.5.4.4.1 Executing the job

You can execute the job from the Designer or from SAP NetWeaver BW.

To execute the job from the Designer, right-click the job name in the project area and click *Execute*.

To execute the job from SAP NetWeaver BW, first configure the connection in BW as follows.

1. Create a new InfoPackage.
2. On the *3rd Party Selections* tab, refresh the view to test the connection to the Data Services RFC server interface.
   If the connection is successful, the *3rd Party Selections* tab displays four returned parameters.
3. Enter the *Repository* and *Jobname*.

   The JobServer and Advanced parameters are for advanced use; leave them blank.
4. Refresh the display again.
   The tab now displays any global variables that have been configured for the job so you can provide values for them.

If you have Data Services Designer or the Administrator open when the job runs, the trace log appears automatically and remains open and active while the job is processing just as it does for a job whose execution is initiated from Data Services.

12.5.4.4.2 Monitoring the job

You can monitor the jobs and view logs in both Data Services and SAP NetWeaver BW. If you launch the job from BW and the Data Services Designer is running, the log opens automatically as it does when you launch jobs from Data Services.

To view the execution logs in Data Services, in the Administrator, under *SAP Connections*, click *RFC Server Interface* and click the name of the interface to view its most recent log. You can also view the previous two logs.

To view the execution logs in SAP NetWeaver BW, access the InfoPackage. In the *Extraction* monitor, click the *Non-SAP System Extraction Log* icon. The log will indicate whether the job completed successfully or if there were execution errors.

Related Information

Management Console Guide: Administrator, Viewing RFC server interface logs [page 1937]

12.6 Connecting to SAP Master Data Services

SAP Data Services lets you create a datastore to the SAP Master Data Services application and import tables for use as sources in Data Services jobs. SAP Master Data Services is powered by the SAP HANA platform.

With the SAP Master Data Services interface, you can use the Designer to:

- Browse and import SAP Master Data Services tables.
- Use SAP Master Data Services tables as sources and targets.

12.6.1 SAP Master Data Services datastores

SAP Data Services uses datastore connections to link with other applications or databases.

- In a design environment, use a datastore to browse or import metadata that represents external tables and other database objects.
- In a production environment, Data Services uses datastore information to move data between source and target databases and applications.

After defining an SAP Master Data Services datastore in Data Services, you can browse tables that you want to import. You can also import tables by name.

Related Information

Defining an SAP Master Data Services datastore [page 2528]
Browsing and importing metadata [page 2529]

12.6.1.1 Defining an SAP Master Data Services datastore

With an SAP Master Data Services datastore connection, you can browse SAP Master Data Services application metadata.

12.6.1.1.1 To define an SAP Master Data Services datastore

1. In the object library of the Designer, go to the *Datastores* tab
2. Right-click inside the object library window and select *New*.
3. In the *Datastore Editor* window, enter a name for this datastore.
   The name can contain alphanumeric characters and underscores. It cannot contain spaces.
4. In the *Datastore type* list, select *SAP Master Data Services*.
5. Enter the connection information.
6. Click *OK*.
   The SAP Master Data Services datastore appears in the object library.

12.6.1.2 Browsing and importing metadata

After you create an SAP Master Data Services datastore, you can browse and import the metadata from the connected application.

12.6.1.2.1 To browse and import metadata

1. Right-click the SAP Master Data Services datastore name and select *Open*, or double-click the datastore name.

   The workspace displays the Tables folder.
2. To view all of the tables, expand the Tables folder.
3. To import table data, right-click one or more tables and select *Import*.

You can also import SAP Master Data Services metadata by right-clicking a Master Data Services datastore in the object library and selecting *Import by name*.

Once tables are imported, you can view column names, data types, and business names for use in creating jobs.

12.6.1.2.2 Table type

SAP Master Data Services applications use the following table types:

- *read:* Can be used as sources.
- *read_write:* Can be used as sources and targets.
- *write_only:* Can be used as targets.

12.7 Loading Spatial Data into SAP HANA

Data Services supports spatial data (for example, point, line, polygon, collection, or a heterogeneous collection) for the following databases:

- Oracle—reading
- SAP HANA—reading and loading

When you import a table with spatial data columns into Data Services, the spatial type columns are imported as character-based large objects (clob). The column has an attribute, Native Type, which has the value of the actual data type in the database; for example, SDO_GEOMETRY for Oracle or ST_GEOMETRY for SAP HANA.

Limitations

- Because spatial columns are imported into Data Services as clob, creating template tables with spatial types is not supported.
- Spatial data cannot be manipulated inside a data flow because the spatial utility functions are not supported.

12.7.1 To load spatial data from Oracle to SAP HANA

1. Import a source table from Oracle to Data Services.
2. Create a target table in SAP HANA with the appropriate spatial columns.
3. Import the SAP HANA target table into Data Services.
4. Create a data flow with an Oracle source as reader, include any necessary transformations, and then add the SAP HANA target table as a loader. Make sure not to change the data type of spatial columns inside the transformations.
5. Build a job that includes the data flow and run it to load the data into target table.

For more information about SAP HANA spatial data support, see the SAP HANA documentation.

12.8 Reference Information

This section contains SAP-specific reference information about SAP objects, functions, and data types.

For more information on all objects, see the *SAP Data Services Reference Guide*.

12.8.1 Objects

12.8.1.1 Common Data Services objects

The following table lists names and descriptions of SAP Data Services objects that are associated with SAP Applications and SAP NetWeaver Business Warehouse.

Object	Class	Description
Datastore	Single-use	Describes the connection information needed for Data Services to access a database, an SAP Applications system, or an SAP NetWeaver BW system. Cannot be dropped.
File format	Reusable	Indicates how data is arranged in a source or target file.
Job	Reusable	A job is a set of objects that you can execute together.
Source	Single-use	An object from which Data Services reads data in a data flow or an ABAP data flow.
Table	Reusable	Indicates an external DBMS table for which metadata has been imported into Data Services, or the target table into which data is or has been placed.

Object	Class	Description
		A table is associated with its datastore; it does not exist independently of a datastore connection. A table retrieves or stores data based on the schema of the table definition from which it was created.
Target	Single-use	An object in which Data Services loads extracted and transformed data in a data flow.

Related Information

Reference Guide: Objects [page 832]

12.8.1.1.1 Datastore

Class

Reusable

Access

In the object library, click the *Datastores* tab.

Description

A datastore provides a connection to a database. Data Services uses the connection defined by the datastore to import descriptions of the database and its tables (metadata). When you specify tables, hierarchies, or other objects as sources or targets in a data flow, Data Services uses the datastore to determine how to read data from or load data to those tables.

In addition, some transforms and functions require a datastore name to qualify the tables they access.

Data Services provides the following datastore types to connect to SAP systems:

- SAP Applications to read data from or load data to an SAP Applications system
- SAP BW Source to read data from an SAP NetWeaver BW system. The options for SAP BW Source datastores are the same as for SAP Applications.
- SAP BW Target to load data to an SAP NetWeaver BW system. The options for SAP BW Target datastores are common to all SAP datastores.

Note that if you delete a datastore from the object library, you must remove references to the datastore from the following locations:

- Source or target tables using this datastore in your data flows.
- The lookup and key_generation functions and Key_Generation, History_Preserving, Table_Comparison, and SQL transform references.
- ABAP data flows for SAP Applications connections.

SAP datastores have the following common options:

SAP option	Possible values	Description
Main window		
Datastore name	Alphanumeric characters and underscores	The name of the object. This name appears on the object in the object library and in the calls to the object.
Datastore type	SAP Applications SAP BW Source SAP BW Target	Select the type of datastore to which you are connecting.
Application server	Computer name, fully qualified domain name, or IP address	Name of the remote SAP application computer (host) to which the software connects.
User name	Alphanumeric characters and underscores	Enter the name of the account through which the software accesses the SAP application server.
Password	Alphanumeric characters and underscores, or blank	Enter the user's password. Note If you have problems connecting to a system that uses SAP_BASIS version 6.40 and earlier, the system might be expecting an uppercase password. To prevent this issue, install the appropriate kernel patch as described in *SAP Note 792850*, "Preparing ABAP systems to deal with incompatible passwords."
Locale		
Language	SAP-supported ISO two-letter language codes or <default>	Select the login language from the possible values in the drop-down list. The <default> option sets the language to what is set in the SAP system user profile.
Code page		Specify the character encoding of character data in the source in the following situations: • For ABAP data flow processing in SAP R/3 4.x systems • For processing IDoc files

SAP option	Possible values	Description
		Also see *Reference Guide: Locales and Multi-byte Functionality*.
SAP		
Client number	000-999	The three-digit client number. Defaults to 800.
System number	00-99	The two-digit system number. Defaults to 00.
Routing string	Refer to the requirements of the application	Enter the SAP routing string used to connect to SAP systems through SAPRouters.
Use sapnwrfc.ini	Yes, No	Select to use an sapnwrfc.ini file, which overrides the datastore settings. Place the sapnwrfc.ini file in the current directory of the process being executed (%LINK_DIR/bin). Defaults to *No*.
Destination name	Refer to the requirements of the application	If using an sapnwrfc.ini file, enter the destination name to reference.
Load balance		
Load balance	Yes , No	Select *Yes* to enable load balancing.
MS host	Computer name, fully qualified domain name, or IP address	Specify the message server host name. Overrides the setting in sapnwrfc.ini.
MS service	Refer to the requirements of the application	Specify this parameter only if the message server does not listen on the standard service sapms<SysID> or if this service is not defined in the services file and you need to specify the network port directly. Overrides the setting in sapnwrfc.ini.
Server group	**`<User input>`**, PUBLIC, SPACE	Optionally specify the group name of the application servers. Default: PUBLIC. Overrides the setting in sapnwrfc.ini.
System ID	Refer to the requirements of the application	Name of the SAP system. Overrides the setting in sapnwrfc.ini.
Authentication		
Authentication	Password, SNC	*Password*: The default. Deploys the user name and password entered in the datastore main window. *SNC*: Select to enable secure network communications (SNC) security and complete the remaining options in this section.
SNC library	Full file path and name of SNC security library	Enter the full path and name of the third-party security library to use for SNC communication (authentication, encryption, and signatures).Overrides the setting in sapnwrfc.ini. Alternatively you can configure the environment variable SNC_LIB.

SAP option	Possible values	Description
SNC name of Data Services	Refer to the requirements of the application	Enter the SNC name that the SAP system uses to identify Data Services. Overrides the setting in sapnwrfc.ini.
SNC name of SAP system	Refer to the requirements of the application	Enter the SNC name of the SAP system for this connection. Overrides the setting in sapnwrfc.ini.
SNC quality of protection	Max available Authentication Integrity Privacy	*Max available*: Maximum quality of protection that is supported by the target SAP system. This value is configured in the SAP Application Server profile parameter snc/data_protection/max. It could be configured to be Authentication, Integrity, or Privacy. *Authentication*: With this protection level, the system verifies the identity of the communication partner, which in this context is Data Services. This is the minimum protection level offered by SNC. No actual data protection is provided. *Integrity*: With this protection, the system detects any changes or manipulation of the data that might have occurred between the two end points of a communication. *Privacy*: With this protection, the system encrypts the messages being transferred to make eavesdropping ineffective. Privacy protection also includes integrity protection of the data. This is the maximum level of protection provided by SNC. Overrides the setting in sapnwrfc.ini.

SAP Applications

The following table lists the options specific to the SAP Applications datastore. The options change depending on which *Data transfer method* you select.

SAP Applications option	Possible values	Description
SAP		
ABAP execution option	Generate and Execute Execute Preloaded	Select the job execution strategy. Your choice affects the required authorizations. *Generate and Execute:* The ABAP created by the job resides on the same computer as the Job Server and is submitted to SAP using the SAP RFC_ABAP_INSTALL_AND_RUN function or /BODS/RFC_ABAP_INSTALL_AND_RUN function. Select this option if the job changes between scheduled executions.

SAP Applications option	Possible values	Description
		Execute Preloaded: ABAP resides on the SAP application server and is submitted using Data Services RFC function modules. Select this option if the job does not change between scheduled executions.
Execute in background (batch)	Yes, No	Specify whether the generated ABAP programs created by SAP application data flows defined with this datastore will execute in batch mode on the SAP server. Batch mode operation is slower than the normal console mode; however, choose batch mode if the application is too long to run during the console mode time frame. Defaults to No.
Target host	Computer name, fully qualified domain name, or IP address	If you chose to execute ABAP programs in the background, specify the target computer (host).
Job class	A, B, C	If you chose to execute ABAP programs in the background, specify the job class.
Data transfer method	Direct download RFC Shared directory FTP Custom transfer	Define how to retrieve data from the SAP application server to the Data Services server: *Direct download:* Use the SAP WS_DOWNLOAD function to stage the data to the *Local directory*. The Job Server reads from the file, then deletes it. Direct download is not recommended for production use because it cannot handle large amounts of data. *RFC*: Use to stream data from the source SAP system directly to the Data Services data flow process using RFC. For secure data transfer, configure *SNC* authentication with the required SNC quality of protection (in the datastore *Authentication* options). *Shared directory:* Both SAP Data Services and SAP have direct access to the directory where data is stored. This method works best when both systems are on Windows and security is not an issue. Use NFS (shares one network drive or directory). *FTP:* Use FTP. *Custom transfer:* Use a third-party program. When you select a data transfer method, the appropriate options for that method appear below the option.
ABAP RFC destination	SAPDS or **`<Destination name>`**	For the *RFC* data transfer method, enter a TCP/IP RFC destination. You can keep the default name of SAPDS and create a destination of the same name in the source

SAP Applications option	Possible values	Description
		SAP system, or you can enter a destination name for an existing destination.
Working directory on SAP server	Directory path or click *Browse*	A directory on the SAP application server where the software can write intermediate files. All the files used by the data flow should be placed in this directory. This directory also stores the transport file used by the FTP, shared-directory, and custom-transfer data transfer methods. By default, the value in this field uses the value that was typed into the *Application server* field. For example, if the value **`sap01`** was typed into the Application server field, the value of *Working directory on SAP server* becomes \\sap01\\.
Custom transfer	These options are visible if you selected the *Custom transfer* data transfer method.	
Custom transfer local directory	Directory path or click *Browse*	The client-side directory to which data from the SAP server downloads.
Custom transfer program executable	Refer to the requirements of the application	The name of the third-party file transfer program you want to use to transfer files from the SAP working directory to the local directory.
Custom transfer user name	Refer to the requirements of the application	(optional) Login ID for the SAP application server to which the custom transfer program connects.
Custom transfer password	Refer to the requirements of the application	(optional) Password for the SAP application server to which the custom transfer program connects. Passwords entered into this option are encrypted.
Custom transfer arguments	Refer to the requirements of the application	(optional) Specify arguments for your custom transfer program. Arguments can add security or compression mechanisms to your program or include Data Services system variables.
Application path to the shared directory	Directory path or click *Browse*	If you selected the *Shared directory* data transfer method, indicate the path from the Data Services server to the SAP application server's working directory. By default, the value in this field uses the value that was typed into the *Application server* field. For example, if the value **`sap01`** was typed into the Application server field, the value of *Working directory on SAP server* becomes \\sap01\\.
Local directory	Directory path or click *Browse*	If you selected the *Direct download* or *FTP* data transfer method, select a client-side directory to which data from the SAP server downloads.

SAP Applications option	Possible values	Description
Generated ABAP directory	Directory path or click *Browse*	Indicate the directory into which the software writes generated ABAP files. It can be the same directory as the local/direct download directory. The path is relative to the Job Server.
Security profile	Refer to the requirements of the application	Specify the security profile you want to use in the generated ABAP program. The user of the SAP datastore must have the required profile.
Routing string	Refer to the requirements of the application	Enter the SAP routing string used to connect to SAP systems through SAProuters.
Number of connection retries	Positive integer	The number of times Data Services tries to establish a connection with the SAP application server. Defaults to 1.
Interval between retries (sec)	Positive integer	The time delay in seconds between connection retries. Defaults to 10.
Upload attributes		
Status	P - SAP Standard Production Program K - Customer Production Program S - System Program T - Test Program	Indicates whether the program is a test program, a system program, or a production program. Default is *T - Test program*. The parameter can have only the value code or the value code and description, separated by a space.
Application	Refer to the drop-down list for available options	Indicates the application area to which the program belongs (Basis, General Ledger, Sales, and so on). The default value is *S - Basis*. The parameter can have only the value code or the value code and description, separated by a space.
Development class (Package)	Refer to the requirements of the application	Indicates the name under which related objects in the ABAP Workbench are grouped together in a package. Default is $TMP. The program is created as a local (non-transportable) object.
Request ID	Refer to the requirements of the application	Indicates the Change and Transport System (CTS) request ID. The default value is blank. This option is populated by Data Services if a non-local program object is created in SAP.
Task ID	Refer to the requirements of the application	Indicates the CTS task ID. The default value is blank. This option is populated by Data Services if a non-local program object is created in SAP.
FTP	These options are visible if you selected the *FTP* data transfer method.	

SAP Applications option	Possible values	Description
FTP relative path to the SAP working directory	Directory path or click *Browse*	Indicate the path from the FTP root directory to the SAP server's working directory. When you select FTP, this directory is required.
FTP host name	Computer (host) name, fully qualified domain name, or IP address	Must be defined to use FTP.
FTP user name	Alphanumeric characters and underscores	Must be defined to use FTP.
FTP password	Alphanumeric characters and underscores, or blank	Enter the FTP password.

SAP Master Data Services

The following table lists the options specific to the SAP Master Data Services datastore. For more information about all other options, see the SAP HANA database datastore section of the *SAP Data Services Reference Guide.*

SAP Applications option	Possible values	Description
SAP Master Data Services		
Schema owner	Refer to the SAP Master Data Services documentation	Enter the schema owner name for foundation tables (the name is usually apps).

Related Information

Authorizations for Data Services [page 2423]
Data Services system variables to transfer a file from SAP applications [page 2480]
Using the sapnwrfc.ini file [page 2439]
Reference Guide: Objects, Database datastores, SAP HANA [page 895]

12.8.1.1.2 File format

Class

Reusable

Access

In the object library, click the *Formats* tab.

Description

A file format describes the structure of an ASCII file. A file format can be specific to a single file, or it can be a generic description used for many data files. Construct file formats from the properties found in the file format editor.

The built-in file format for SAP is the Transport_Format which is under the *Flat Files* node. The following options are for the SAP transport type only.

Property	Possible values	Description
General		
Type	SAP transport	The format of the data in the text file. Available properties change based on the selected file format type.
Name	Any alphanumeric character and underscores (_)	A descriptive name for the file format. This name appears in the object library.
Data File(s)		
Location	Local, Job Server	(Optional) The computer on which the data file(s) are stored.
File name(s)	File name, file name including full path name, or blank	In new and edit modes, specify an existing file on which you base the file format description. Data from this file appears in the Column Attributes area. In these modes, you can leave this property blank. In source and target modes, specify the location of the actual file for this source or target. In these modes, you cannot leave this property blank. For added flexibility, you can enter: ● A variable that is set to a particular file with full path name. ● A list of files, separated by commas, or a file name containing a wild card. In this case, Data Services reads all these files as a single source.
Delimiters		
Column	Tab, Semicolon, Comma, Space	Specify the character that indicates the end of one column and the beginning of the next. Alternatively, enter an ASCII character. Valid ASCII characters range from /0 to /254.

Related Information

Reference Guide: Objects, File format [page 920]

12.8.1.1.3 Job

Class

Reusable

Access

- In the object library, click the *Jobs* tab.
- Right-click a job in the project area.

Description

A job is a set of objects that you can execute together. For Data Services to execute the steps of any object, the object must be part of a job. A single job must include all the steps you want executed.

The following are the additional references to SAP applications.

- Data Services retrieves results from the following types of steps:
  - ABAP data flows
  - ABAP data files
- Jobs have properties that determine what information Data Services collects and logs when running the job. Use Trace properties to select the information that Data Services monitors and writes to the trace log file during a job. You can turn several SAP application-related traces on and off.

Trace	Description
ABAP Query	Writes a message when an ABAP program is submitted and when the ABAP program completes. Also writes the ABAP job status every 15 seconds.
IDoc file reader	Writes a message when an IDoc file reader starts and when it finishes. This trace also reports: ○ The name of the file that the reader opens ○ The file status (succeeded or failed) ○ Total number of IDocs processed from the file

Trace	Description
RFC Function	Writes messages when Data Services calls an RFC function or BAPI. Specifically, this trace writes messages when: ○ An RFC function or BAPI starts ○ Processing for an individual record starts ○ Processing for an individual record finishes ○ New connections to SAP applications are established during real-time jobs ○ An RFC function or BAPI finishes
SAP Table Reader	Writes messages when Data Services reads from an SAP application table source. Specifically, this trace writes messages when: ○ Data Services establishes a connection with SAP ○ Data Services begins reading from the SAP application table ○ The RFC call to retrieve table content starts ○ The RFC call to retrieve table content finishes ○ Data Services finishes reading from the SAP application table

Related Information

Reference Guide: Objects, Log [page 935]

12.8.1.1.4 Source

Class

Single-use

Access

- To insert an SAP application table as a source, open the object library, go to the *Datastores* tab, select the table, drag it into a batch or ABAP data flow in the workspace, and select *Make Source*.
- To insert a file as a source, open the object library, go to the *Formats* tab, select a flat file format for the file, drag it into the workspace, and select *Make Source*. Use the file format editor to specify the file name.
- To insert an Open Hub table as a source in a data flow, select the table from the local object library *Datastores* tab and drag it into the work space.
- To insert an SAP ODP source as a source in a data flow or an ABAP data flow, select it from the local object library *Datastores* tab, drag it into the work space, and modify the import options as necessary.

- To insert an ABAP data flow as a source, open the object library, go to the *Data flows* tab, select the ABAP data flow, drag it into a batch or ABAP data flow in the workspace.
- To insert an XML message or file as a source, open the object library, go to the *Formats* tab, select the DTD or XML Schema format for the XML file, drag it into the workspace, and select *Make XML message source* or *Make XML file source*. Use the source editor to specify the test file name for the message or the source file name for the file.
- To view options of tables or files, click the name of the table or file in the workspace or in the project area. This opens the table or file editor.
- To view options of ABAP data flows, right click the data flow in the workspace or project area, select *Properties*, and go to the *Options* tab.
- To insert an IDoc message or file as a source, open the object library, click the *Datastores* tab, click an SAP Applications datastore, click the IDoc category, select an IDoc, drag it into a data flow, and select *Make IDoc message source* or *Make IDoc file source*. Use the source editor to specify object attributes.

> i Note
>
> A real-time job source can be one and only one XML or IDoc message. If another message source is already in the data flow definition, Data Services will not allow you to place the IDoc in the workspace.

Description

A source is an object from which Data Services reads data.

In a batch job, a source can be a table, an SAP application table, a previously defined template table, a file, an XML file, an IDoc file, ABAP data flow, or an SAP ODP source.

In a real-time job, a source can be a table, an SAP application table, a previously defined template table, a file, an IDoc file, an XML file, an IDoc message, or an XML message. Each real-time job must have exactly one real-time data source—either an IDoc message or an XML message.

In an ABAP data flow, a source can be another ABAP data flow, an SAP application table, a file located on the SAP application server, or an SAP ODP source. Files must be in a file format that SAP can read. Data Services provides a predefined flat file format called Transport_Format that SAP can read.

You can select an SAP application table as a source in any data flow. However, when you use an SAP application table as a data flow source, Data Services extracts data using the Data Services-provided RFC (remote function call) /BODS/RFC_STREAM_READ_TABLE. If this function is not loaded to the SAP application server, Data Services extracts data using the SAP-supplied function, RFC_READ_TABLE. This function call limits extracted data to 512 bytes per row.

When you use an SAP application table as a source in an ABAP data flow, Data Services extracts data using ABAP. For large amounts of data (larger than a few rows), performance is generally better when you extract data from SAP using an ABAP data flow.

With SAP ODP sources, Data Services automatically detects whether the extractor uses the ODP data replication API, which provides optimal performance and changed-data capture (delta) capability. In a regular data flow Data Services uses RFC to call the ODP data replication API. In an ABAP data flow, Data Services generates the ABAP that calls the ODP data replication API.

IDocs are used as message sources in real-time jobs and as file sources in batch or real-time jobs. Data flows in real-time jobs that use IDocs as message sources can be processed one at a time or in parallel. IDoc message sources are generated directly from SAP.

You can configure an IDoc file source in a data flow of a batch job to read files in groups and use variable file names.

> Note
>
> IDocs cannot be used as source or target objects in ABAP data flows.

A table source has options in addition to the datastore information:

Option	Description
Join rank	Indicates the rank of the table relative to other tables joined in a data flow. The software joins tables with higher join ranks before it joins tables with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.
Cache	Indicates whether Data Services should read the required data from the table and load it into memory. Because an inner table of a join must be read for each row of an outer table, you may want to cache a table when it is used as an inner table in a join and when the data retrieved will fit into available memory. Best practice is to specify the cache only in the Query transform editor. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. The default option in the Query editor is Automatic. The automatic setting in the Query editor carries forward the setting from the source table. Note If any one input schema has a cache specified in the Query editor, the Data Services Optimizer considers only Query editor cache settings and ignores all source editor cache settings. For non-SAP application tables, there are two options: • *Yes*: The table is always cached unless it is the outer-most table in a join. • *No*: The table is never cached. For SAP application tables in an ABAP data flow, there are three options: • *Yes*: The table is always cached unless it is the outer-most table in a join. • *No*: The table is never cached. • *Automatic*: Data Services determines whether to cache the table. The default option is *No*. When two SAP application tables are involved in an inner join and you cache both tables, Data Services improves performance by generating ABAP using the array operations SELECT FOR ALL ENTRIES and IN_MEMORY MERGE JOIN. These operations are exclusive to inner joins, specifically two-way inner joins. Data Services sorts then merges the inner and outer tables to optimize join performance.

Option	Description
	To enable these operations, set the cache option for both tables to *Yes*. The result is that both inner and outer tables are read into internal tables prior to the join. Use *Package size* on the outer loop of the join to limit the size of internal tables.
Package size	Indicates the maximum number of rows loaded into memory. Limiting rows reduces memory consumption. Package Size is an extension of the SELECT_INTO_TABLE operation, which uses one SELECT statement to retrieve blocks of rows to create an internal table. Use this option when caching a table that might consume excessive memory. Data Services does not limit rows when you enter 0, a negative number, or when you set *Cache* to *No*.
Array fetch size	Indicates the number of rows to load into the ABAP memory table before Data Services fetches the rows. Avoids exceeding memory capacity on the source side. Default is 1000.
Execute in background (batch)	If the SAP table reader is taking more time to execute than what is in the SAP system dialog work process setting rdisp/max_wprun_time (SAP RZ11 transaction), select this option to avoid exceeding the time limit. Ensure an *RFC destination* has been specified in the datastore editor for this source. If there is no RFC destination defined, create an RFC destination object (SAP transaction sm59) and enter the name in the datastore definition.

Note

Data Services automatically uses the SELECT_INTO_TABLE operation when you cache a database table and the content of the cache populates an internal table, or when you cache one or more LOOKUP functions against one or more tables and the contents of the LOOKUP cache populates an internal table.

A file source also has the *Join Rank* and *Cache* options in addition to the file format information. Data Services interprets *Join Rank* and *Cache* the same in a file as a table.

An XML message source has these options in addition to read-only XML Schema or DTD format information:

Option	Description
XML test file	The location relative to the Job Server of an XML-formatted file to use as the message source when you execute the job in test mode.
Enable validation	A check box to turn on the comparison of the incoming message to the stored XML Schema or DTD format. When this option is selected, the real-time job throws an exception if the incoming message is not valid. When you are developing a real-time job, this validation helps you to make sure sample data is both valid and well-formed. If you select this option in production, make sure to include appropriate error handling either in the Data Services job or the client application to process an error caused if a data flow in the real-time job receives data that does not validate against the imported format.
implied: Join rank	The XML message source is always the outer table in a join. You cannot change its join rank. This option is implied and does not appear in the editor.

An XML file source has these options in addition to read-only XML Schema or DTD format information:

Option	Description
XML file	The location relative to the Job Server of an XML-formatted file to use as the source.
Enable validation	A check box to turn on the comparison of the incoming data to the stored XML Schema or DTD format. When this option is enabled, the data flow throws an exception if the incoming source is not valid.
implied: Join rank	The XML file source is always the outer table in a join. You cannot change the join rank for it. This option is implied and does not appear in the editor.

An IDoc message source in a real-time job has the following options:

Option	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc test file	Browse the network to find a test file to read.
IDoc Type	(Read-only) Data Services will only process IDoc files (listed in the IDoc Test File box) if they are of this IDoc type. For example, SOPGEN01. This information is entered when you import metadata for the IDoc Type.

An IDoc file source, in a real-time job, has the following options

.

Option	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc File/s	Browse to a network file that will contain IDocs generated in SAP. Data Services reads the contents of this file/s when this data flow runs.
IDoc Type	(Read-only) Data Services will only process IDoc files (listed in the IDoc File box) if they are of this IDoc type. For example, SOPGEN01. This information is entered when you import metadata for the IDoc Type.
Join Rank	Indicates the rank of the table relative to other tables joined in a data flow. The software joins tables with higher join ranks before it joins tables with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.
Include file name column	Determines whether to include a column in the source output that contains the source IDoc file name. Defaults to *No*. Change the value to *Yes* when you want to identify the source IDoc file in situations such as the following: ● You specified a wildcard character to read multiple source IDoc files at one time ● You load from different source IDoc files on different runs

Option	Description
Modify	If the file name is included, this button enables you to modify *File name column* and *Column size*.
File name column	If the file name is included, the name of the column that holds the source IDoc file name. Defaults to *DI_FILENAME*.
Column size	If the file name is included, the size (in characters) of the column that holds the source file name. Defaults to *100*. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.
Include path	If the file name is included, determines whether to include the full path name of the source IDoc file. Defaults to *No*.

Related Information

IDoc sources in batch jobs [page 2494]
Designer Guide: Embedded Data Flows [page 391]
IDoc file [page 2562]

12.8.1.1.5 Table

Class

Reusable

Access

In the object library, click the *Datastores* tab.

Description

You can use a table from a non-SAP Applications datastore as a source or target in a data flow. You cannot use a table from a non-SAP Applications datastore in an ABAP data flow.

You can use an SAP application table as a source or target in an ABAP data flow or as a source in a data flow. You cannot use an SAP application table as a target in a data flow.

Tables have the following built-in attributes:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the table.
Business_Name	Business name imported with table metadata.
Table_Usage	How the object is used when it was accessed most recently. Expect values such as Source, Target, Lookup Table, or Comparison Table.
Total_Number_Of_ Rows_Processed	The number of rows loaded into the table in the last successful load.
Date_last_loaded	The time the table was last successfully loaded by Data Services.
Number_Of_Rows_ Rejected	The number of rows rejected in the last successful load.
Number_Of_Inserts	The number of rows inserted in the last successful load.
Number_Of_Updates	The number of rows updated in the last successful load.
Date_Created	The date that the object was created.
Estimated_Row_ Count	A configurable estimate of the table size used in calculating the order tables are read to perform join operations. Used for SAP application tables only.
Number_Of_Deletes	The number of rows deleted in the last successful load.
Elapsed_Time_For_ Load	The time it took to load this table in the last successful load.
Table_Type	The type of datastore object. Most tables are type Table.
SAP_Table_Class_ Name	The table class name as designated by the SAP_Table_Class value.
SAP_Table_Class	Table type information. Used for SAP application tables only. 1 — Transparent table 2 — INTTAB 3 — Cluster

Attribute	Description
	4 — Pool 5 — View 6 — Append (Structure)

12.8.1.1.6 Target

Class

Single-use

Access

- To display target options, click the name of the target in the workspace or in the project area. This opens the object editor.
- To display target properties, right click a target and choose *Properties*.

Description

A target is an object in which Data Services loads data.

In a batch job, a target can be a flat file, a table or template table, an XML file, an IDoc file, an IDoc message, or an ABAP data flow.

In a real-time job, a target can be a file, a table, a template table, an XML file, an XML message, an IDoc file, or an IDoc message.

In an ABAP data flow, a target can be a file or a data transport object.

Target files

You can use any flat file format in a batch or ABAP data flow as a target. You can use any DTD or XML Schema format for XML file and XML message targets.

If the schema defined in the format doesn't match the schema that is input to the target, Data Services provides validation errors to identify the mismatch.

Target tables

No tables can be used in ABAP data flows as targets.

InfoSource or DataSource target

The following table contains option and description information specific to SAP NetWeaver BW InfoSource and SAP NetWeaver BW DataSource target tables.

See the *SAP Data Services Reference Guide* for option information common to all target tables.

Option	Description
Options	
Number of loaders	Loading with one loader is known as "single loader loading". Loading when the number of loaders is greater than one is known as "parallel loading" The default number of loaders is 1. You can specify any number of loaders. When parallel loading, each loader receives the number of rows indicated in the *Rows per commit* option, and applies the rows in parallel with the other loaders. For example, if you choose a *Rows per commit* of 5000 and set the *Number of Loaders* to 3: in parallel, the loaders each receive a maximum of 5000 rows; then each loader performs the necessary processing and, when complete, sends the data in a packet to the SAP NetWeaver BW system. Due to differences in processing, the loaders may not send the packets to the SAP NetWeaver BW system in sequential order. Note Parallel loading is not supported for the hierarchy type.

Target Idocs

IDoc messages can be sent from Data Services on a regular schedule or in real-time. Data Services allows you to use a batch or real-time job to configure IDoc message targets. Any number of IDoc targets can be included in a single data flow.

IDoc targets support reduced message types. When Data Services detects a reduced message type, it retrieves the appropriate metadata from SAP and creates a reduced IDoc, removing unused segments and giving unused fields the value of '/'.

Note

Reduced message type functionality is only available for some message types (e.g., DEBMAS, CREMAS, etc.).

The attributes of an IDoc target depend on the type of target and the type of data flow. To see the IDoc attributes, double-click an IDoc target object in a data flow diagram.

- IDoc message targets in a data flow of a batch job

Attribute	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc Type	(Read-only) Displays the IDoc Type you imported Data Services into for this object. For example, SOPGEN01.
Partner Number	This is an SAP term for where to send the IDoc in the SAP application. Enter the partner number of the IDoc you will be processing as a message target in Data Services. If you do not know the partner number, consult your SAP administrator.
Partner Type	This is an SAP term for an attribute of a partner number. Enter the partner type. For example, `LS` for logical system.
Message Type	This is an SAP term for an EDI message for this IDoc Type. Enter the message type.
Batch size	Specifies the number of IDoc messages to accumulate before Data Services sends the IDoc messages to the SAP server. This attribute improves performance because Data Services sends multiple IDoc messages with each transmission. Data Services sends the accumulated IDoc messages even if the number of seconds is less than the value in *Batch wait timeout*.
Do not create TID	This attribute improves performance by not creating a transaction ID for SAP transmissions. However, transaction services (such as re-transmission and checking) are not done.
Batch wait timeout	This attribute limits the wait time (in seconds) before Data Services sends the IDoc messages to the SAP server. A value of `0` specifies to wait indefinitely. This attribute improves performance because Data Services sends multiple IDoc messages with each transmission. Data Services sends the accumulated IDoc messages even if the number of messages is less than the value in *Batch size*.

- IDoc message targets in a data flow of a real-time job

Attribute	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc Test File	Browse to a network file that will contain IDocs generated in Data Services. For example: `C:/temp/`**`<filename>`**. Use this location to test the output of the IDoc message target before starting a real-time job in the Data Services Administrator. Data Services writes to this file when this real-time job is executed in test mode.
Partner Number	This is an SAP term for where to send the IDoc in SAP. Enter the partner number. If you do not know the partner number, consult your SAP administrator.

Attribute	Description
Partner Type	This is an SAP term for an attribute of a partner number. Enter the partner type. For example, `LS` for logical system.
Message Type	This is an SAP term for an EDI for this IDoc Type. Enter the message type.
Batch size	Specifies the number of IDoc messages to accumulate before Data Services sends the IDoc messages to the SAP server. This attribute improves performance because Data Services sends multiple IDoc messages with each transmission. Data Services sends the accumulated IDoc messages even if the number of seconds is less than the value in *Batch wait timeout*.
Do not create TID	This attribute improves performance by not creating a transaction ID for SAP transmissions. However, transaction services (such as re-transmission and checking) are not done.
Batch wait timeout	This attribute limits the wait time (in seconds) before Data Services sends the IDoc messages to the SAP server. A value of `0` specifies to wait indefinitely. This attribute improves performance because Data Services sends multiple IDoc messages with each transmission. Data Services sends the accumulated IDoc messages even if the number of messages is less than the value in *Batch size*.
IDoc Type	(Read-only) Data Services will only process IDoc files (listed in the IDoc Test File box) if they are of this IDoc type. For example, SOPGEN01. This information is entered when you import metadata for the IDoc Type.

- IDoc file targets in data flows

Attribute	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc File	Browse to a network file that will contain IDocs generated in Data Services. Data Services writes to this file when the data flow is executed. When you configure an IDoc File target, Data Services allows you to capture data for more than one IDoc. Each time a root segment passes into a file target (for example *IDoc File* `C:\temp\idoc`), a new file is created and named using an eight character suffix starting with `00000000.txt`. For example an *IDoc File* set to `C:\temp\idoc` could store `idoc00000000.txt` to `idoc99999999.txt`.
IDoc Type	(Read-only) The IDoc type you imported from SAP into Data Services for this object. For example, SOPGEN01.

Related Information

Reference Guide: Objects, Target [page 960]

12.8.1.2 SAP-specific objects

This section describes SAP-specific objects available in Data Services.

Object	Class	Description
ABAP data flow	Reusable	Extracts data from SAP application sources.
Custom ABAP transform	Reusable	Inserts custom ABAP logic blocks into a generated ABAP program for an ABAP data flow.
Data transport	Single-use	Identifies a file to hold a data set extracted by an ABAP data flow.
ODP source	Single-use	An ODP source is a source object that Data Services uses to read data from an SAP application.
Hierarchy	Reusable	Browse, search, and import hierarchy metadata.
IDoc file	Reusable	An intermediate document that SAP produces to move data between servers. In Data Services, an IDoc file can be used in a real-time or batch job as a source or a target.
IDoc message	Reusable	An intermediate document that SAP produces to move data between servers. In Data Services, an IDoc message can be used in a real-time job as a source or a target and in a batch job as a target.
Open Hub Table source	Single-use	A source object that Data Services uses to read data from an SAP NetWeaver Business Warehouse system.
Transport_Format	Reusable	The Transport_Format is a Data Services predefined file format object for reading flat files in SAP applications. It is automatically used to define data transport objects added to ABAP data flows.

Related Information

The Transport_Format [page 2502]
File format [page 2538]

12.8.1.2.1 ABAP data flow

Class

Reusable

Access

- To insert an existing ABAP data flow, go to the object library *Data Flows* tab, open the list of ABAP data flows, select the ABAP data flow you want to add, and drag it into the workspace.
- To insert a new ABAP data flow, select the ABAP data flow icon in the tool palette, and click in the workspace.

Description

An ABAP data flow extracts data from SAP application sources. You can insert an ABAP data flow into any other data flow (ABAP, batch, or real-time). An ABAP data flow produces a data set you can use as a source in other data flows.

You can define parameters to pass values into the ABAP data flow. If you cannot express the required extraction logic using the Data Services interface, you can embed ABAP logic into an ABAP data flow by creating a new, custom ABAP transform.

When Data Services executes ABAP data flows, it translates extraction requirements into ABAP programs and passes them to SAP to execute. An ABAP data flow can store the extracted data set in an SAP application table or in text file saved on the SAP application server. To use the data outside the ABAP data flow, save the data in a file and use a data transport object to pass the data to the parent data flow. The data transport object defines a file format and indicates how data is transferred from the SAP application server to Data Services.

When you use the ABAP data flow as a source in a join (either with other ABAP data flows, SAP sources, or with non-SAP sources), you can assign a join rank to the source. When joining sources, Data Services reads sources with higher join ranks before reading those with lower join ranks. Set join rank in the ABAP data flow properties when you first define the ABAP data flow.

> Note
>
> Join rank must be a positive integer. Zero is the default value. With a zero join rank, Data Services determines join order.

An ABAP data flow can contain the following objects:

- Sources
- Targets
- Transforms (Custom ABAP)
- Queries
- ABAP data flows

ABAP data flows have the following options on the *Options* tab.

Option	Description
Datastore	Specifies the datastore corresponding to the SAP application tables or files this ABAP data flow accesses.
Generated ABAP file name	Specifies the file name containing ABAP program code that Data Services generates from this SAP ABAP data flow. This file is written to the directory specified in the datastore definition.
ABAP program name	Specifies the name Data Services uses for the ABAP program name that runs in SAP ABAP. It must begin with Y or Z.
Job name	Specifies the name used for the job that runs in SAP. It defaults to the name of the data flow.
ABAP row limit	Limits the number of rows that will be read by this ABAP data flow. Use this option for testing purposes to limit the time that the parent batch job will run. This option can also be used to limit the number of rows read using the Designer's Data Scan feature. ABAP data flows return one file of data from scans. Individual objects inside an ABAP data flow cannot be read using Data Scan.
Join rank	Indicates the rank of the table relative to other tables joined in a data flow. The software joins tables with higher join ranks before it joins tables with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.
Parallel process threads	Specify the number of threads for parallel processing, which can improve performance by maximizing CPU usage on the Job Server computer. For example, if you have four CPUs, enter 4 for this option.
Cache	Indicates whether Data Services should cache the data set produced by this ABAP data flow for use in other operations. If the result is part of another ABAP data flow, the cache option is ignored. Best practice is to specify the cache only in the Query transform editor. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. The default option in the Query editor is Automatic and in a table source is No. The automatic setting in the Query editor carries forward the setting from the source table. Note If any one input schema has a cache specified in the Query editor, the Data Services Optimizer considers only Query editor cache settings and ignores all source editor cache settings. *Yes*: The table is always cached unless it is the outer-most table in a join. *No*: The table is never cached. *Automatic*: Data Services determines whether to cache the table.

Option	Description
	The default option is *No*.

ABAP data flows have the following properties on the *General* tab:

Attribute	Description
Name	The name of the object. This name appears in the object library and in calls to the object.
Always execute during recovery	Indicates that Data Services should re-execute this ABAP data flow during an automatic recovery when it is part of a re-executed work flow that is a recovery unit. In this case, Data Services re-executes the ABAP data flow even if it completed successfully in an earlier run.
Description	Your description of the ABAP data flow.

Related Information

Creating custom ABAP transforms [page 2462]
Adding an ABAP data flow [page 2452]

12.8.1.2.2 Custom ABAP transform

Class

Reusable

Access

- In the object library, click the *Transforms* tab then right-click and select *New ABAP Transform*. After renaming your custom ABAP Transform, drag it into an ABAP data flow. From the *Tools* menu, select *Variables* to define parameters.
- You can reuse an existing transform by dragging and dropping the object from the *Transforms* tab.

Description

A custom ABAP transform executes a piece of ABAP logic to produce a set of rows to be used by subsequent transforms in the same ABAP data flow.

A custom ABAP transform does not take any input sets and produces one and only one output data set. It allows for any number of input parameters, but no output parameters.

A custom ABAP transform has the following properties:

Properties	Description
Name	Specifies the name for the custom ABAP program that appears in the object library and in diagrams. This name cannot contain spaces.
Description	Allows you to document the transform's operation.

A custom ABAP transform has the following options:

Option	Description
ABAP language file name	Indicates the name of the file containing the ABAP program to execute.
Join rank	Indicates the rank of the output data set relative to other data sets included in a join. The software joins data sets with higher join ranks before it joins data sets with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For more information, see the *Data Services Reference Guide*.
Output schema	Lists the columns in the schema produced by this ABAP program. To add a column to the schema, right-click and choose *New Column*.
Column type	Indicates the data type of the selected column. This option appears after you create a column, click outside the column name, and then select the column.

12.8.1.2.3 Data transport

Class

Single-use

Access

- To insert, open an ABAP data flow diagram in the workspace, select the data transport icon in the tool palette, and click on the workspace.

- To view options, click the name of the data transport. This opens the data transport editor.

Description

Data transports are targets in ABAP data flows. A data transport takes the data SAP extracts during an ABAP data flow and stores it in a file defined by the Transport_Format file format.

Use a data transport when you want to use the extracted data in a data flow outside the ABAP data flow. In this case, you must use the ABAP data flow as a source in the parent data flow.

Do not use a data transport when you want the extracted data kept in a file on the SAP application server or in an SAP table. In that case, use a file or SAP table as the target in the ABAP data flow.

The data transport editor shows the schema produced by the ABAP data flow and the options that define the data transport. These options specify the location of the file and indicate whether to append to or overwrite the data in the file.

You can set the delimiter SAP uses to write data to the file. The default delimiter is a tab character. Change the delimiter in the Transport_Format file format definition. Valid ascii character delimiters range from /0 to /254.

The following table lists the options in the data transport editor:

Data transport editor option	Description
File Name	The name of the saved staging file.
Working directory on SAP	The location of the saved staging file. This is the *Working directory on SAP server* specified for the datastore associated with the current ABAP data flow.
Append file	Adds the data set to the existing contents of the file.
Replace file	Clears the file before writing the data set to it.

Data transports have one attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

12.8.1.2.4 ODP source

Class

Single-use

Access

To insert an SAP ODP source into a data flow, open the local object library, select the *Datastores* tab, select the ODP source from the SAP Applications datastore, and drag it into the workspace.

Description

An SAP ODP source is an object that SAP Data Services uses to read data from an SAP application.

Data Services automatically detects whether the ODP source uses the ODP data replication API, which provides optimal performance and changed-data capture (delta) capability. Typically you use ODP source extractors in a regular data flow, where Data Services uses RFC to call the ODP API. However if you want to do additional processing such as lookups and joins or for more advanced applications such as debugging or push-down optimization, you can use them in an ABAP data flow, where Data Services generates the ABAP that calls ODP API.

ODP source option	Description
Name	Name of the ODP source as defined in the datastore.
Type	Identifies the ODP source type as defined by SAP. Examples: TRAN: Transaction ATTR: Attribute
Method	Identifies the extraction type as defined by SAP. Examples: V: Transparent table or database view F1: Function module (complete interface) F2: Function module (simple interface)
Datastore	The SAP Applications datastore name.
Datastore type	The type of SAP datastore (SAP Applications).
Application server	Name of the remote SAP application computer to which Data Services connects.
Language	Select the login language from the possible values in the drop-down list. You can enter a customized SAP language in this option. For example, you can type S for Spanish or I for Italian.
Client number	The three-digit SAP client number. Default is 800.
System number	The two-digit SAP system number. Default is 00.
Join rank	Indicates the rank of the source relative to other tables and files in the data flow when creating a join. The software joins sources with higher join ranks before it joins sources with lower join ranks.

ODP source option	Description
Package size	Limits the number of rows brought into memory when building internal tables. By limiting the number of rows, less memory is used to construct internal tables.
Initial load	For changed-data capture (CDC), indicates whether to reload all the data. Select to return a full extraction; clear to return only changed data. If you change the filter properties for an ODP source, “reset” the job by running it with the source option *Initial load* selected. Then you can run the job with CDC filters applied (with *Initial load* cleared).
Extract from datetime	For changed-data capture (CDC), optionally enter a global variable of type datetime that defines from when to extract changed data. If the datetime value is the same as or before the value from the last execution, Data Services repeats the changed-data extraction. If the datetime value is later than the value from the last execution, Data Services returns the new data. Example: Yesterday the job ran with a datetime value of 2010.10.28 03:00:00. To reload the same data again (for example because of a problem with the execution), use the same datetime value again. To get new changes, use a later value, for example 2010.10.29 03:00:00.
Parallel process threads	For ODP sources that use the Operational Data Provider data replication API (ODP data replication API), specify the number of threads used to read the data. For example, if you have four CPUs on your Job Server computer, enter the number 4 for this option to maximize performance.

Related Information

Reading from SAP ODP sources [page 2442]

12.8.1.2.5 Hierarchy

Class

Reusable

Access

In the object library, click the *Datastores* tab and expand a datastore listing (click the plus sign next to the datastore name).

Description

Hierarchy objects allow you to browse, select, and import hierarchical information from a source database. Hierarchy objects can import the following hierarchical information:

Source application	Hierarchies extracted
SAP application	Data Services represents SAP application sets as hierarchies. In SAP applications, sets are group of objects such as cost element groups or cost center groups. The sets allow you to see an individual object or all similar objects in a related area. Hierarchy objects can import set information from SAP applications basic or single-dimension sets. In particular, you can import cost element and cost center hierarchies.

To extract hierarchical information, select a hierarchy type, import its metadata into your repository, then drag the hierarchy icon from the object library into a data flow.

SAP application hierarchies

External view

Hierarchy types are listed in the external browsing view for an SAP Applications datastore. From this view, you can choose to import a hierarchy type from a specific controlling area, or you can import a hierarchy type regardless of its association with a controlling area.

You can also import a hierarchy by specifying the hierarchy type by name. Select the datastore in the object library, right-click and choose *Import By Name*.

Imported hierarchy type

After you import the hierarchy type, it appears in the object library nested under the datastore name.

The hierarchy object has the following properties:

Property	Description
Name	The name of the hierarchy object. The name is constructed as follows: `set_table-hierarchy_group(datastore)`
Description	The description of the hierarchy type as included in the SAP application table.
Controlling_Area	The controlling area from which you imported the hierarchy type. You could specify the controlling area by selecting the hierarchy type in the controlling area view or by entering the controlling area name in the Import by Name window when you imported the hierarchy type. If you did not specify a controlling area, the value in Controlling_Area is an asterisk (*).

Property	Description
	If you import a chart of account hierarchy type, the controlling area will be inferred. The Controlling_Area value will never be NULL.
Set_Table	The table from which the hierarchy data is extracted.
Chart_Of_Account	The chart of account from which you imported this hierarchy type. This value is NULL for many hierarchy types associated with controlling areas but not chart of accounts.
Hierarchy_Group	The hierarchy type imported.
Check_Table	The location of the range of values for the hierarchy type leaf nodes.
Domain_Name	The data type and range definitions of the leaf node values.

Selecting the hierarchy in the object library and choosing *Open* displays the object properties and the source schema that results from the hierarchy extraction. Each row of the hierarchy object output describes a single relationship in the hierarchy.

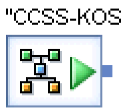
Column name	Description
Parent_id	The parent node in the relationship described by this row.
Parent_desc	A text description of the parent node.
Child_id	The child node in the relationship described by this row.
Child_desc	A text description of the child node.
Value_from	The start of the range of values associated with a leaf node. The range is found in the check table, which is recorded in the hierarchy object attributes. NULL if the row does not describes a leaf node.
Value_to	The end of the range of values associated with a leaf node. The range is found in the check table, which is recorded in the hierarchy object attributes. NULL if the row does not describes a leaf node.
Leaf_flag	A value of 1 indicates that the child node is a leaf node. A value of 0 indicates that the child node is not a leaf node.
Root_flag	A value of 1 indicates that the parent node is the root node. A value of 0 indicates that the parent node is not the root node.
Tree_level	The level of the child node in the hierarchy relative to the root node.
Cont_area	The controlling area associated with the hierarchy type. If this hierarchy type was selected without specifying a controlling area, the value of Cont_area is an asterisk (*).
Chart_of_acct	The chart of account associated with the hierarchy type.
Set_table	The table from which the hierarchy data is extracted.

Note

For every root node in the hierarchy, the hierarchy object produces a row with NULL as the root node. Depending on the final use of the hierarchy data, you may want to filter the NULL parent rows out of the hierarchy data.

Hierarchy instance

You can drag a hierarchy from the object library into an ABAP data flow definition. The hierarchy object appears in the data flow definition as follows:



The hierarchy editor displays the datastore information for the hierarchy and provides options for the instance of the hierarchy object:

Option	Description
All root nodes or Root node	Indicates that all hierarchies (root nodes) will be extracted from the specified hierarchy type. Turn off this option to specify a single hierarchy to extract. Look in the external view of the datastore to determine specific hierarchy names.
All controlling areas or Controlling area	Indicates that all hierarchy types for all controlling areas in which the groups appear will be extracted. Turn off this option to specify a single hierarchy type for a single controlling area to extract. ℹ Note For best performance, make sure to specify a single controlling area.

Related Information

Designer Guide: Datastores, Importing metadata through a database datastore [page 219]

12.8.1.2.6 IDoc file

Class

Reusable

Access

In the object library, click the *Datastores* tab, click an SAP Applications datastore, and click the list of available IDocs.

Description

IDoc file sources are handled the same way in data flows of real-time and batch jobs.

A real-time job that contains an IDoc file source processes its files one at a time. A real-time job that contains an IDoc message source can process messages one at a time or in parallel.

Multiple File Read in batch jobs

One or more IDoc files can be saved into one file or into many individual files to be processed at a later time.

When you run a real-time job in Test mode, Data Services can read multiple IDoc files at one time. To configure Data Services to read files in groups, list the names of these IDoc files or use one name with wildcard characters in the *IDoc Files* box of the source editor in the Designer.

The following list provides three examples:

- If you specify the IDoc file `D:\temp\IDOC_SAP*.txt`, Data Services will process data from all files in the D:\temp directory with names beginning with IDOC_SAP and the extension .txt.
- If you specify the IDoc file `D:\temp\IDOC_SAP?.txt`, Data Services will process data from all files in the D:\temp directory with:
  - names beginning with IDOC_SAP
  - any character before the extension
  - the extension `.txt`.
- If you specify the IDoc file `D:\temp\IDOC_01.txt`, `D:\temp\IDOC_02.txt`, `D:\temp\IDOC_03.txt`, Data Services will process data from these three IDoc files.

You can identify the source file for each row by using the Include file name column.

IDoc file sources are validated against the IDoc Name, its related hierarchy, and the SAP release.

Variable File names in batch jobs

You can also use variable file names in the *IDoc File/s* box. Define a global or local variable for an initialization script of a job. For example,

```
$SOPGEN01_<filepath> =    “D:\temp\IDOC_SAP*.txt”
```

Then enter **`<$filepath>`** in the *IDoc Files* box. This way you can avoid drilling down to each IDoc object to configure the location of source files.

Related Information

Source [page 2541]

Target [page 2548]
IDoc message [page 2564]

12.8.1.2.7 IDoc message

Class

Reusable

Access

In the object library, go to the *Datastores* tab, open the SAP Applications datastore, and open the list of available IDocs.

Description

An IDoc is an intermediate document that an SAP application produces to communicate transaction data between SAP application servers. SAP applications offers numerous IDocs that describe different transaction types. The data in an IDoc supports a particular transaction. An IDoc can contain data from multiple tables and can contain hierarchical data.

You can use an IDoc message as a source or target in a real-time job. You can use an IDoc message as a target in a batch job. You cannot use an IDoc in an ABAP data flow.

You can use the multiple file read and variable file name features with IDoc message sources in real-time jobs. However, these features can only be used when the job is run in test mode.

IDoc message sources must be configured in the Data Services Administrator as well as in Data Services Designer.

Real-time jobs that contain an IDoc message source can be processed one at a time or in parallel. To enable parallel processing, select *Administrator* > *Real-time* > *Client Interface* in the Data Services Administrator.

The *Parallel Processing* option allows you to increase the number of IDoc source messages processed per minute for the IDoc type specified. This option is disabled by default.

The parallel processing option allows the Access Server to send an IDoc to a service queue (where it waits for a service provider) and continue with the next IDoc without waiting for reply. The maximum number of outstanding IDoc requests in the queue is the number of IDocs received or 4, whichever is smaller.

> **Note**
>
> Where a strict IDoc processing sequence is required, do not use the *Parallel Processing* option.

To use IDocs, you must first import the metadata from the SAP application to your repository. Data Services imports the schema for the IDoc, maintaining hierarchical relationships among the data fields. You can display the IDoc schema by double-clicking the IDoc name in the object library.

When IDoc metadata is imported into Data Services or when IDocs are sent and received, the control data appears as the first segment for all IDoc objects. The control data is commonly prefixed with EDI_DC. The control data contains IDoc administrative information, such as IDoc number and sender or receiver information.

Related Information

Importing IDoc metadata using the Search option [page 2441]
Source [page 2541]
Target [page 2548]
IDoc file [page 2562]
Creating real-time jobs using IDoc sources [page 2496]

12.8.1.2.8 Open Hub Table source

Class

Single-use

Access

To insert an Open Hub source into an SAP Data Services data flow, open the local object library, go to the *Datastores* tab, select the Open Hub table from the SAP BW Source datastore, and drag it into the workspace.

Description

An Open Hub Table is a source object that SAP Data Services uses to read data from an SAP NetWeaver Business Warehouse system.

> Note
>
> You can only use an Open Hub Table as a source object, and it can only be used in a regular data flow. You cannot use it in an ABAP data flow.

The following options are available for an Open Hub source.

Open Hub source option	Description
Make Port	Makes the source table an embedded data flow port. For more information, see “Embedded Data Flows” in the *SAP Data Services Designer Guide*.
Join rank	Indicates the rank of the source relative to other tables and files in the data flow when creating a join. The software joins sources with higher join ranks before it joins sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in the source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For more information, see “Other Tuning Techniques” in the *Data Services Performance Optimization Guide* .
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. There are two options: ● *Yes*: The source is always cached unless it is the outer-most source in a join. ● *No*: The source is never cached. The default is *No*. A cache setting specified in the Query transform editor FROM tab overrides any cache setting specified in a source. Best practice is to specify the cache setting only in the Query transform editor. Note For data flows that you created prior to version 11.7, the default cache type is in-memory. If the data retrieved does not fit into available memory, change the cache type to pageable in the data flow *Properties* window.
Open Hub Table	Specifies the name of the Open Hub table to read from. The default value is the name of the current Open Hub table. You can do any of the following: ● Leave the default value if different people are not using it simultaneously. ● Select another Open Hub table name from the drop-down list, but the one you select must have the same schema as the current Open Hub table. ● Select a substitution parameter from the drop-down list, but the one you select must have the same schema as the current Open Hub table. You must define the substitution parameter before creating this data flow. This capability allows you to reuse the data flow to read from another Open Hub destination that obtains its

Open Hub source option	Description
	data from the same InfoProviders with the same schema. For more details, see the Related Topics below.
Execute process chain before reading	Specifies whether or not to execute the process chain in the BW system before this Data Services data flow reads data from the Open Hub table. The process chain reads data from the BW InfoProvider objects and loads the Open Hub table from which the Data Services Open Hub source object reads the data. By default, this option is not selected. In this case, you must start the process chain with the function sap_openhub_processchain_execute in a script.
Process chain name	Specifies the name of the process chain in the BW system for this Open Hub table when *Execute process chain before reading* is selected. If you select *Execute process chain before reading*, the software fills in this option with the first process chain in the attributes for the current Open Hub table. Ensure that this is the appropriate process chain. If you change the name of the Open Hub table, you can select the corresponding process chain name from the drop-down list. You can specify a substitution parameter that you define before creating this data flow.
Parallel SAP Connections	Specifies the number of Open Hub connections to read multiple data packets from a BW system simultaneously. The default value is 0. Valid values are 0 and any positive integer. A value of 0 or 1 disables parallel reading. You can specify a substitution parameter that you define before creating this data flow.

Related Information

Creating a data flow for multiple users to read one InfoProvider [page 2519]

12.8.2 Functions

SAP Data Services does not support functions that include tables as input or output parameters except RFC and BAPI functions imported from SAP Applications.

12.8.2.1 SAP application RFC and BAPI function calls

You can call SAP application Remote Function Call (RFC) enabled functions, including Business Application Programming Interface (BAPI) functions, from queries inside data flows.

To make an RFC function available to call from Data Services data flows, import the metadata for the function from the SAP application server using an SAP Applications datastore connection. Be aware that the requirements for RFCs and BAPIs, and therefore their metadata, may be different between versions of SAP applications.

If you design data flows with BAPI calls against one version of an SAP application, then change datastores to a later version of SAP, Data Services allows this without the need to reimport the BAPI. Any new parameters added to the function call, including additional columns in table parameters, are added automatically to the call and filled with NULL values. Thus Data Services allows you to design jobs that are portable between SAP systems.

For a Data Services job to execute an RFC function, the login indicated by the datastore into which you imported the function must include the appropriate permissions required to execute the functions.

After you import the metadata for an SAP function, the function is listed in the Functions category of the SAP Applications datastore. You will also see the function in the function wizard listed under the datastore name.

When a function can provide a single output value, you can call the function in a column mapping in a query. Data Services provides a function wizard (available by clicking the *Functions* button on the *Where* tab of a query) to help you include input parameters for the function call. Input parameters can include expressions.

When a function can provide more than one output value, you can call the function by including it as an object in the output schema of a query. Data Services provides a function wizard (available through a short-cut menu) to help you include input and output parameters for the function call.

Data Services supports tables as input and output parameters for SAP RFC and BAPI functions. The function import process automatically includes the metadata for tables included as function parameters.

To specify a table as an input parameter to a function, the table must be an input to a query, either as a top-level input or nested under the top-level. The table must also be available in the FROM clause of the context where you call the function. Data Services maps columns in the input schema by name to the columns in the table used as the function input parameter. You need only supply the columns that are required by the function. At validation, if Data Services encounters type mismatches between supplied columns and the function signature, it attempts to convert the given type to the expected type. For type mismatches that it cannot resolve, Data Services produces validation errors.

One of the values that an RFC function can return is AL_RFC_RETCODE. This column contains a flag that identifies the success or failure of the function call. The possible values for AL_RFC_RETCODE are as follows:

Value	Description	Returned by
BOBJ_DI_RFC_OK	The RFC call succeeded. This value is replaced by the return value from the RFC call.	Data Services
BOBJ_DI_RFC_ENABLE_ERROR	The function is not RFC enabled.	Data Services
BOBJ_DI_RFC_NOT_ENABLED	The function is not RFC enabled.	Data Services

Value	Description	Returned by
BOBJ_DI_RFC_CALLRECEIVE_ERROR	The function call in SAP returned an error.	Data Services
BOBJ_DI_RFC_R3_CONN_EXCEPTION	The SAP Applications datastore connection cannot be created, because of a connection error, invalid user, password, system number, or host name.	Data Services
BOBJ_DI_RFC_CALL_ERROR	The connection completes, but the call fails in SAP.	Data Services
BOBJ_DI_RFC_GET_RESULT_ERROR	Data Services cannot obtain the result of the function call from SAP.	Data Services
BOBJ_DI_RFC_COMMIT_ERROR	Data Services cannot commit the work because the BAPI_TRANSACTION_COMMIT call returned an error.	Data Services
BOBJ_DI_RFC_BAPI_NOT_SAME	The Data Services repository contains a different BAPI function than in SAP.	Data Services
RFC_OK	The function call succeeded. Look for the results or errors that it returns.	SAP application
RFC_FAILURE	The function call returned an error. If the function is a BAPI, details for the cause of the error are available in the RETURN structure available as an output from the function.	SAP application
RFC_EXCEPTION	The function call returned an error. If the function is a BAPI, details for the cause of the error are available in the RETURN structure available as an output from the function.	SAP application
RFC_SYS_EXCEPTION	The function call returned an error and closed the connection to Data Services. If the function is a BAPI, details for the cause of the error are available in the RETURN structure available as an output from the function.	SAP application
RFC_CALL	The function call was received by SAP. If this value is left, the function failed to return a success flag after starting.	SAP application
RFC_INTERNAL_COM	An internal communication error occurred within SAP.	SAP application
RFC_CLOSED	The SAP application closed the connection and cancelled the function call.	SAP application

Value	Description	Returned by
RFC_EXECUTED	The SAP application already executed the function call.	SAP application
RFC_MEMORY_INSUFFICIENT	The SAP application does not have enough memory available to process the function call.	SAP application
RFC_NO_TID	The transaction ID is not available to SAP.	SAP application
RFC_RETRY	The SAP application did not process data yet. SAP will retry the function call.	SAP application
RFC_NOT_FOUND	The SAP application could not find the function.	SAP application
RFC_CALL_NOT_SUPPORTED	The SAP application does not support the function call.	SAP application
RFC_NOT_OWNER	The login in the Data Services datastore cannot connect to SAP.	SAP application
RFC_NOT_INITIALIZED	The Data Services RFC library did not initialize properly.	SAP application
RFC_SYNCHRONIZE	The SAP application is busy processing a synchronous call.	SAP application
RFC_SYSTEM_CALLED	Data Services is busy executing a call from SAP.	SAP application
RFC_VERSION_MISMATCH	The version of the function call from Data Services is incompatible with the function expected by SAP.	SAP application

The RETURN structure for BAPIs varies between releases of SAP applications:

Field	Description
TYPE	blank — success S — success E — error W — warning I — information A — abort
CODE	Error message numbers

Field	Description
MESSAGE	Error message text in the language chosen at login

This TYPE value is blank or NULL depending on the current setting of the Server option *Convert SAP null to null*. Check this option by choosing ▶ *Tools* > *Options* ◢ in the Designer. In particular when calling BAPI functions, the data you provide through the BAPI call might be different from the data that you use to test a BAPI directly in the SAP GUI interface. The SAP application interface automates data handling, where the BAPI operation undercuts the interface level.

Consider the following issues:

- All character values must be uppercase
- Padding values
- Assumed decimal values (QTY)
- Codes are language-specific
- Automatic type conversion
- SAP application version-specific behavior

To determine the data requirements of various SAP application functions, you can read the function requirements in the SAP GUI transaction screens:

- BAPI list by functional area: `bapi`
- BAPI and RFC source and input and output parameters: `se37`

You can also determine appropriate values, such as the language-specific code values, by looking at the table where the data is ultimately stored.

Related Information

Designer Guide: Real-time Jobs, Calling application functions [page 390]
Reference Guide: Data Types, Type conversion [page 1039]

12.8.2.2 Data Services built-in functions

This section describes the Data Services functions that are specific to SAP.

Function	Category	Description
lookup	Lookup	Finds a value in one table or file based on values in a second table or file.
sap_extractor_delta_initialize	SAP	Sets a marker from which to return changed data.
sap_openhub_processchain_execute	SAP	• Starts the process chain that extracts data from an InfoProvider (InfoArea, InfoCube, or DataStore object) on SAP NetWeaver Business Warehouse and

Function	Category	Description
		loads the extracted data into an Open Hub Destination table. ● Monitors the process chain status and the Open Hub Destination request notification.
sap_openhub_set_read_status	SAP	Sends the read status for the Open Hub table to SAP NetWeaver BW. A read status causes SAP NetWeaver BW to release the Open Hub Destination table so that other users can access it.
substr	String	Returns a specific portion of a string starting at a given point in the string.
sy	Miscellaneous	Returns the value of an SAP system variable at run time. This function is only available through query transforms in ABAP data flows.

12.8.2.2.1 lookup

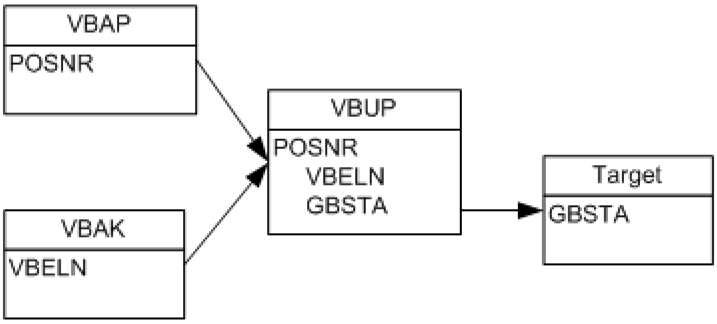
For SAP application tables, no owner value is required, but the periods are. For example: `sap_ds..KNA1`

`<cache_spec>`: The caching method the lookup operation uses. List within single quotes.

With regard to the three possible settings:

Setting	Description
NO_CACHE	Reads values from the lookup_table for every row without caching values.
PRE_LOAD_CACHE	For SAP application tables, applies the filters and loads the result_column and the compare_column into an internal SAP table. The function reads the values from the internal table. Select this option if the number of rows in the table is small or you expect to access a high percentage of the table values.
DEMAND_LOAD_CACHE	Select this option if the number of rows in the table is large and you expect to access a low percentage of the table values frequently. For SAP application tables, loads the result_column and compare_column into an internal SAP table when two or more lookups use the same translation table. So, Data Services creates a single SELECT statement to load the combined values from all the lookups into an internal SAP table. Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.

You can use the lookup function when retrieving data from SAP application tables in an ABAP data flow. For example, suppose you are joining two tables, VBAK and VBAP, and you want to retrieve a value from a third table, VBUP.



You can create a lookup function with two **<expression>** and **<compare_column>** pairs:

```
lookup(sap_ds..VBUP, GBSTA, 'none', 'NO_CACHE', VBELN, VBAK.VBELN, POSNR,
VBAP.POSNR)
```

This function returns the value from the GBSTA column (the **<result_column>**) in the VBUP table (the **<lookup_table>**) that corresponds to the VBELN value (first **<compare_column>**) in the VBAK table (first **<expression>**) and the POSNR value in (second **<compare_column>**) the VBAP table (second **<expression>**). When no corresponding value is found, the function returns "none."

When there are multiple matching rows in the **<lookup_table>** and the lookup table is an SAP application table, the lookup function randomly selects a matching row and returns the value in the result_column for that row. To avoid random selection:

- Use the lookup_seq function if the table has a sequence column that discriminates between matching records
- Use an outer join and add criteria to the WHERE clause to discriminate between multiple matching records

12.8.2.2.2 sap_extractor_delta_initialize

Syntax

For SAP ODP sources using changed-data capture, use this function in a script to indicate to the ODP data replication API to only extract changed data (delta) for subsequent executions.

```
sap_extractor_delta_initialize (<datastore_name>, <extractor_name>, <$g1>)
```

where $g1 is a sample name for a global variable of type string. After calling the function, $g1 is empty to indicate success; otherwise, it displays E for error.

12.8.2.2.3 sap_openhub_processchain_execute

Performs the following tasks:

- Starts the process chain that extracts data from an InfoProvider (InfoArea, InfoCube, or DataStore object) on SAP NetWeaver Business Warehouse and loads the extracted data into an Open Hub Destination table.

- Monitors the process chain status and the Open Hub Destination request notification.

When the function returns successfully, an Open Hub table source in SAP Data Services can then read the data from the Open Hub Destination table.

Note

You can only use this function in a script. It is not valid in a query or audit object.

Below is the function syntax as a reference. The function wizard is explained in the next section.

Syntax

```
sap_openhub_processchain_execute('<datastore>','<open_hub_table>',
'<process_chain>',<$logid>,<$ReturnTxt>)
```

Where

`<datastore>`	Specifies the datastore name. You can specify either a constant string or a substitution parameter. The data type is varchar(256). Note The maximum length depends on the Data Services repository type. For most repository types the maximum length is 256, for MySQL the length is 64, and for MS SQL server the length is 128.
`<open_hub_table>`	Specifies the Open Hub Destination table. You can specify either a constant string or a substitution parameter. The data type is varchar(30).
`<process_chain>`	Specifies the name of the process chain that extracts data from the InfoProvider in SAP NetWeaver BW and loads the data to the Open Hub Destination table. You can specify either a constant string or a substitution parameter. The data type is varchar(30).
`<$logid >`	(Optional) Specifies a variable to obtain a value that depends on the function return value (see "Relationship between return value and value of logid variable" below). The required variable data type is varchar(25).
`<$ReturnTxt>`	(Optional) Specifies a variable to retrieve the description of the return status of the process chain.

	The required variable data type is varchar, and you can define the length you want for this variable.

Return value

varchar(1)

Returns one of the following values.

Return value	Description
B	Open Hub Destination is being read by another user.
E	Data Services error while executing the function.
R	Process chain execution failed with errors in BW system.
X	Process chain execution has been canceled in BW system.
S	Function successfully executed the Open Hub extraction Data Transfer Process (DTP) and received extraction request notification.

Relationship between return value and value of logid variable

The value of the logid output variable depends on the function return value, as the following table shows.

Return value	$logid variable value	$ReturnText variable value	Action
B	Process chain log ID of the other user that is currently reading the Open Hub Destination	Status of current Open Hub extraction	Either wait and try again or stop executing the data flow that contains the Open Hub Destination table.
E	Data Services error log number	Data Services error text	Stop executing the data flow that contains the Open Hub Destination table, and analyze the Data Services error.
R	Your process chain log ID	Error from process chain	Stop executing the data flow that contains the Open Hub Destination table, and use the log ID in the BW system to see the detail state of the process chain error.
X	Your process chain log ID	Error from process chain	Stop executing the data flow that contains the Open Hub Destination table, and use the log ID in the BW

Return value	$logid variable value	$ReturnText variable value	Action
			system to see the detail state of the process chain error.
S	Open Hub extraction request ID	Status of your Open Hub extraction	Use the request ID in the BW system to obtain detail loading statistics (such as number of packets loaded and number of records loaded).

Example

The following sample script commands check the return value, generate an exception, and print the error if the function is not successful.

```
$status = sap_openhub_processchain_execute('open_hub_datastore',
'Materials','Materials_PC',<$lpcogid>,
<$returntxt>);
If ($status != 'S') raise_exception ('Materials_PC process chain execution failed
' || $returntxt);
```

Restrictions

The following are restrictions for using Open Hub Destinations:

- Only one job at a time can read an Open Hub Destination table.
- A process chain of an Open Hub Destination can contain only one of its Data Transfer Processes (DTPs).
- A process chain cannot contain DTPs for more than one Open Hub Destination.

12.8.2.2.3.1 To define an sap_openhub_processchain_execute function

1. To access the function wizard for sap_openhub_processchain_execute from the Script Editor, click *Functions* or *...* at the top of the window.
2. Select *sap_openhub_processchain_execute* from the list of functions.
   The *Define Parameters* window opens.
3. Select an SAP BW Source datastore name from the drop-down list. You can also select a substitution variable from the list. If you type in a datastore name, put single quotes around the name.
4. Select the name of a Open Hub table from the drop-down list. Only the names of the imported Open Hub Tables appear in this list.
5. Select the name of a Process Chain from the drop-down list.
6. Specify a variable that will get the BW log ID for the process chain after the function executes. You must define the variable before you can use it.

7. Specify a variable that will get the description of the status after the function executes. You must define the variable before you can use it.

12.8.2.2.4 sap_openhub_set_read_status

Sends the read status for the Open Hub table to SAP NetWeaver BW. A successful read status causes SAP NetWeaver BW to delete the data from the Open Hub Destination table.

Syntax

```
sap_openhub_set_read_status('<datastore>','<destination>',
<status>,<$returntxt>)
```

Where

`<datastore>`	Specifies the datastore name. You can specify either a constant string or a substitution parameter. The data type is varchar(256). Note The maximum length depends on the Data Services repository type. For most repository types the maximum length is 256, for MySQL the length is 64, and for MS SQL server the length is 128.
`<destination>`	Specifies the Open Hub Destination in the BW system. You can specify either a constant string or a substitution parameter. The data type is varchar(30).
`<status>`	Specifies the read status. Possible values are either a variable or one of the following string constants: ● 'X' for Read Successful ● Any other value indicates that the Read failed. The data type is varchar(1).
`<$returntxt>`	(Optional) Specifies a variable to return the status log of the function. The required variable data type is varchar, and you can define the length you want for this variable.

Return value

varchar(1)

Returns one of the following values.

Return value	Description
S	Success
E	Error

Example

The following sample script commands sends the status of the Open Hub table read to the BW system and prints the status.

```
$status = sap_openhub_set_read_status('BR9", 'PS_BOOK_5', 'X', $returntxt):
print ('Status: ' || $status);
```

12.8.2.2.5 substr

In an ABAP data flow, the first character is position 0; in other data flows, the first character is position number 1. If **<start>** is 0 in a non-ABAP data flow, the new string begins with the first character (that is, position 1). If **<start>** is negative, the new strings begins with the first character in the string (that is, position 0 in an ABAP data flow or position 1 in a non-ABAP data flow).

12.8.2.2.6 sy

Returns the value of an SAP system variable at run time. This function is only available through query transforms in ABAP data flows.

Syntax

Syntax

```
sy('<SAP_variable>')
```

Return value

varchar(255): The value of the SAP system variable. You may need to recast the return value to the actual data type of the system variable in SAP.

Where

<SAP_variable>: A string value containing the name of the SAP system variable. This value is not case sensitive. Enclose the name in single quotation marks (').

When the sy function is executed, Data Services generates the appropriate function call in the ABAP for the ABAP data flow (appends `SY-` to the **<SAP_variable >** that you specify) and returns the result of the function call at run time.

The table SYST in SAP lists the available system variable, their data types, and descriptions.

If the given **<SAP_variable >** does not exist in SAP, Data Services produces a run-time error:

```
ABAP program <Generated ABAP Program> syntax error: <The data object "SY" has no
component called "ABC">.
```

No value is returned when this error occurs.

Example

Functions and results

Function	Results
`sy('SUBRC')`	0 if the subroutine call returned successfully
`sy('mandt')`	The client ID number of the current SAP application client
`sy('UNAME')`	The logon name of the user
`sy('DATUM')`	The current date from the SAP application server
`sy('UZEIT')`	The current time from the SAP application server
`sy('TCODE')`	The current transaction code

12.8.3 Data types

The following sections discuss SAP application data types used in Data Services.

Related Information

Reference Guide: Data types [page 1027]

12.8.3.1 Conversion to/from Data Services internal data types

The following table lists the conversions from SAP application data types to Data Services internal data types.

Source	External data type	Converts to Data Services data type
SAP application	date (D)	date
	time (T)	time
	float (F)	double
	integer (I)	int
	numeric (N)	varchar
	packed (P)	decimal
	char (C)	varchar
	hexadecimal (X)	varchar

The following table lists the conversions from Data Services internal data types to SAP application data types.

Internal data type	Is converted to SAP application data type
date	date (D)
datetime	date (D)
decimal	decimal (P)
double	float (F)
int	int (I)
interval	int (I)
numeric	char (C)
real	float (F)
time	time (T)

Internal data type	Is converted to SAP application data type
varchar	char (C)

12.8.3.2 Working with null values from SAP applications

SAP applications use a different standard for null values than Data Services or the target databases that Data Services supports. You can configure Data Services to account for the difference in one of two ways:

- Always converting the SAP equivalent of a null value to the Data Services NULL constant
- Reading the SAP null value literally

Each method has consequences for how you can use null-value tests in analytic processing. Data in a given warehouse must be loaded over time using the same method.

The default configuration for Data Services is to read the SAP value literally. You can change this configuration by navigating in the Designer to ▶ *Tools* ▶ *Options* ▶ *SAP* ▶ and selecting or deselecting the *Convert SAP null to null* option.

12.8.3.3 Design considerations for SAP application null behavior

When choosing which method to use to handle SAP application null values and when designing extraction logic in Data Services, consider the following issues:

- Primary keys built from concatenated values
  Spaces are often used in SAP applications where null values would cause an error in a primary key built from several key values concatenated together. If you choose to have Data Services automatically convert these spaces to null values, you may have to include null-value replacement logic in your data flows to avoid errors in primary keys for extracted tables.
- Null date values
  If dates are left to Data Services and the target database conventions to determine the conversion of an SAP null date value, the result may not be useful for constructing reports.
- Filters for null values

The value used to match an SAP null value depends on the data type of the value.

Data type	SAP application null value
Strings	single space
Numbers	zero
Dates	`00000000`
Times	`000000`

Related Information

Null dates [page 2582]

12.8.3.4 Null dates

The concept of a null date in SAP applications is represented by a value called INITIAL that is a property of every SAP application data element. You can see INITIAL dates displayed through SAPGUI as `00/00/0000` (month 0, day 0, year 0).

In the default configuration of Data Services (where *Convert SAP null to null* is disabled), INITIAL dates are written to an ABAP data file as NULLs. In ABAP data files, NULLs are represented by empty strings. They are then read by the parent data flow, which converts every empty string to a single space. If passed through again without data type or value change, Data Services converts the single blank into 01-JAN-1900 before loading it into a target.

The single space is not a valid date value. When Data Services encounters a single space in date conversion operations (implicit or explicit) or when evaluating an expression, the results are not predictable. To avoid unpredictable results, the data flow must explicitly convert the INITIAL date to a valid date value.

Consider these techniques to convert INITIAL dates to valid date values:

- Convert INITIAL dates to valid dates using the nvl function in the ABAP data flow
  In an ABAP data flow, Data Services produces ABAP for the nvl function that looks for an INITIAL date rather than a null value. The mapping expression can be the following:

  ```
  nvl(date_column, valid_date)
  ```

  where the valid date is a parameter passed to the data flow or a string indicating the appropriate initial date for your data warehouse.
  An example mapping expression would be:

  ```
  nvl(VBAK.ANGDT, '19901231')
  ```

- Turn on the option *Convert SAP null to null* and use nvl in regular (non-ABAP) data flows
  If an INITIAL date is extracted from SAP and loaded directly into a Data Services transport file, the INITIAL date is converted to a null value. If the Convert SAP null to null option is turned on, the null values in the transport file are brought into the data flow as null values. You can then use the nvl function in the data flow to detect the null date and substitute an appropriate valid date for it.

13 Supplement for Siebel

13.1 Overview of the SAP Data Services Siebel interface

The Data Services Siebel interface allows you to create Siebel datastores and import tables for use as sources in Data Services jobs. With this interface, you can use the Designer to:

- Browse and import Siebel tables grouped by type, business component, and Siebel repository
- Browse to tables in a Siebel repository using the following path through higher level data: ▶ *applications* › *screens* › *views* › *business objects* › *business components* › *tables* ◀.
- Import function metadata

Related Information

Browsing and importing metadata [page 2585]

13.2 System requirements

The Siebel interface allows you to connect SAP Data Services with a third-party database under the Siebel application layer. You can import tables (including table and column descriptions) and function metadata. Data Services also displays information from both the logical business objects and user interface layers of Siebel applications. This makes it easier to find and import tables.

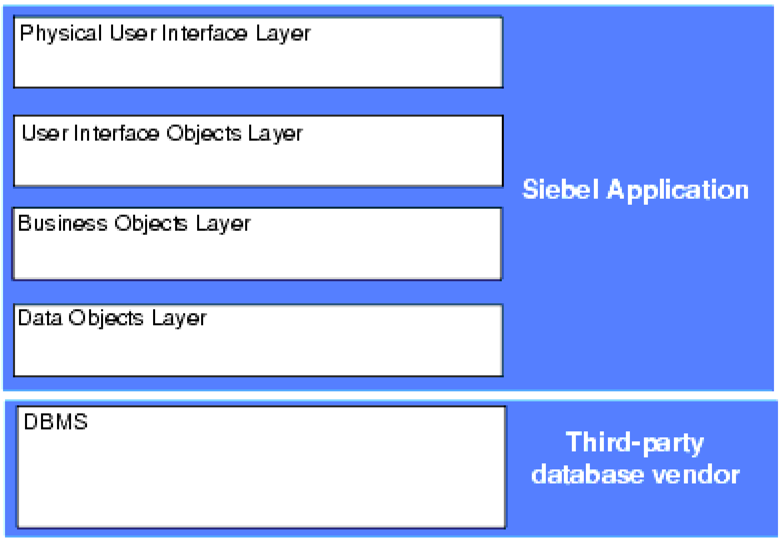

Physical User Interface Layer
User Interface Objects Layer
Business Objects Layer
Data Objects Layer
Siebel Application
DBMS
Third-party database vendor

Figure 10: Siebel Architecture

Install the drivers you need to connect your database with Data Services on the same computers on which you install the Designer and Job Server components.

The Siebel interface supports Siebel version 7.5.2 and higher compatible versions. It can be installed using the Data Services installer.

13.2.1 Security

Because SAP Data Services connects to Siebel applications through a database connection, Data Services can see all Siebel application objects that a database user can access without requiring the application's user login security information.

13.3 Datastores

SAP Data Services uses datastore connections to link with other applications or databases.

- In a design environment, use a datastore to browse or import metadata that represents external tables and other database objects.
- In a production environment, Data Services uses datastore information to move data between source and target databases and applications.

After defining a Siebel datastore in Data Services, you can browse tables by type, or drill down on business components or Siebel repositories to find the tables you want to import. You can also import tables, functions, and business components by name.

13.3.1 Defining a Siebel datastore

With a Siebel datastore connection, you can browse Siebel application metadata.

13.3.1.1 To define a Siebel datastore

1. In the object library of the Designer, go to the *Datastores* tab
2. Right-click inside the object library window and select *New*.
3. In the *Datastore Editor* window, enter a name for this datastore (`DS_SiebelApps`, for example).
4. In the *Datastore type* list, select *Siebel*.
5. Enter the connection information.
6. Click *OK*.
   The Siebel datastore appears in the object library.

13.3.2 Browsing and importing metadata

After you create a Siebel datastore, you can browse and import the metadata from the connected application.

> Note
>
> While you cannot browse functions, you can import them by name.

13.3.2.1 To browse and import metadata

1. Right-click the Siebel datastore name and select *Open*, or double-click the datastore name.

   The workspace displays a list of folders.
2. To find the tables you want to import, expand the Table by type, Business Component, or Repository folder. See the remainder of this section for a description of these folders.
3. To import table data, right-click one or more tables or business components and select *Import*.

   > Note
   >
   > When you import a table into SAP Data Services via the Siebel interface, the software does not preserve the source folder (table hierarchy). All tables display at the same level. For example, import all the tables in a Business Component, then add select statements, joins, and mappings using the Designer to create the same associations used in your Siebel application.

You can also import Siebel metadata by right-clicking a Siebel datastore in the object library and selecting *Import by name*.

Once tables are imported, you can view column names, data types, and business names for use in creating jobs.

13.3.2.2 Table - by type

Siebel applications use many table types. You can browse tables by looking at tables grouped by type.

13.3.2.2.1 Business component

Business components are sets of tables used to create a logical Siebel application object called a business object. Business components are grouped in alphabetical folders.

You can select and import one or more business components or tables.

13.3.2.2.2 Repository

Siebel repositories hold a variety of Siebel application objects. You can drill through them to find the tables you want to import.

These associations apply:

- *Seibel Repository:* Repository
- *Seibel Dynamic Commerce:* Application
- *Auctions:* Screen
- *Auction Items:* View
- *Auction Services:* Business Object
- *Auction Item Detail:* Business Component

13.3.3 Extracting data from Siebel applications

When extracting data from Siebel applications, SAP Data Services processes and converts data types appropriately.

www.sap.com/contactsap