



## Reference Guide

- SAP BusinessObjects Data Services 4.1 Support Package 1 (14.1.1.0)

2012-11-22

## Copyright

© 2012 SAP AG. All rights reserved. SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, SAP HANA and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries. Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company. Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase, Inc. Sybase is an SAP company. Crossgate, m@gic EDDY, B2B 360°, B2B 360° Services are registered trademarks of Crossgate AG in Germany and other countries. Crossgate is an SAP company. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary. These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

2012-11-22

# Contents

<b>Chapter 1</b>	<b>Introduction.....</b>	<b>15</b>
1.1	Welcome to SAP BusinessObjects Data Services.....	15
1.1.1	Welcome.....	15
1.1.2	Documentation set for SAP BusinessObjects Data Services.....	15
1.1.3	Accessing documentation.....	18
1.1.4	SAP BusinessObjects information resources.....	19
1.2	Overview of this guide.....	19
1.2.1	About this guide.....	19
1.2.2	Who should read this guide.....	20
<b>Chapter 2</b>	<b>Objects.....</b>	<b>21</b>
2.1	Characteristics of objects.....	21
2.1.1	Object classes .....	21
2.1.2	Object options, properties, and attributes.....	23
2.2	Descriptions of objects .....	24
2.2.1	Annotation.....	27
2.2.2	Batch Job.....	28
2.2.3	Catch .....	37
2.2.4	COBOL copybook file format .....	41
2.2.5	Conditional .....	48
2.2.6	Data flow .....	49
2.2.7	Datastore.....	52
2.2.8	Document.....	106
2.2.9	DTD.....	106
2.2.10	Excel workbook format .....	116
2.2.11	File format.....	123
2.2.12	Function.....	136
2.2.13	HDFS file format.....	137
2.2.14	Log.....	139
2.2.15	Message function.....	144
2.2.16	Outbound message.....	145
2.2.17	Project.....	145

2.2.18	Query transform.....	146
2.2.19	Real-time job.....	147
2.2.20	Script.....	150
2.2.21	Source.....	151
2.2.22	Table.....	162
2.2.23	Target.....	165
2.2.24	Target Writer migrated from Data Quality .....	209
2.2.25	Template table.....	209
2.2.26	Transform.....	212
2.2.27	Try.....	213
2.2.28	While loop.....	213
2.2.29	Work flow.....	214
2.2.30	XML file.....	216
2.2.31	XML message.....	218
2.2.32	XML schema.....	220
2.2.33	XML template.....	234
<b>Chapter 3</b>	<b>Smart editor.....</b>	<b>237</b>
3.1	Accessing the smart editor.....	237
3.2	Smart editor options.....	238
3.2.1	Smart editor toolbar.....	238
3.2.2	Editor Library pane.....	239
3.2.3	Editor pane.....	239
3.3	To browse for a function.....	242
3.4	To search for a function.....	243
<b>Chapter 4</b>	<b>Data Types.....</b>	<b>245</b>
4.1	Descriptions of data types.....	245
4.1.1	date.....	246
4.1.2	datetime.....	248
4.1.3	decimal.....	248
4.1.4	double .....	249
4.1.5	int (integer).....	249
4.1.6	interval.....	250
4.1.7	Limitations for long and blob.....	250
4.1.8	numeric.....	254
4.1.9	real.....	254
4.1.10	time.....	255
4.1.11	timestamp.....	256
4.1.12	varchar.....	257

4.2	Data type conversion.....	258
4.2.1	Date arithmetic.....	258
4.2.2	Conversion to or from internal data types.....	259
4.2.3	Conversion of data types within expressions.....	283
4.2.4	Conversion among number data types.....	283
4.2.5	Conversion between explicit data types.....	286
4.2.6	Conversion between native data types.....	286
<b>Chapter 5</b>	<b>Transforms.....</b>	<b>291</b>
5.1	Operation codes.....	291
5.2	Descriptions of transforms.....	292
5.3	Data Integrator transforms.....	296
5.3.1	Data_Transfer.....	296
5.3.2	Date_Generation.....	307
5.3.3	Effective_Date.....	309
5.3.4	Hierarchy_Flattening.....	314
5.3.5	History_Preserving.....	323
5.3.6	Key_Generation.....	329
5.3.7	Map_CDC_Operation.....	331
5.3.8	Pivot (Columns to Rows).....	337
5.3.9	Reverse Pivot (Rows to Columns).....	343
5.3.10	Table_Comparison.....	346
5.3.11	XML_Pipeline.....	356
5.4	Data Quality transforms.....	359
5.4.1	Transform configurations.....	359
5.4.2	Downloading blueprints and other content objects.....	365
5.4.3	Dynamic transform settings.....	366
5.4.4	About Data Quality fields.....	368
5.4.5	Associate .....	370
5.4.6	Country ID .....	384
5.4.7	Data_Cleanse.....	386
5.4.8	DSF2® Walk Sequencer.....	415
5.4.9	Geocoder.....	423
5.4.10	Global Address Cleanse.....	444
5.4.11	Global Suggestion List .....	505
5.4.12	Match.....	516
5.4.13	USA Regulatory Address Cleanse.....	577
5.4.14	User-Defined.....	646
5.4.15	Address Cleanse reference.....	654
5.5	Platform transforms.....	695
5.5.1	Case.....	695

5.5.2	Map_Operation.....	698
5.5.3	Merge.....	700
5.5.4	Query .....	701
5.5.5	Row_Generation.....	730
5.5.6	SQL.....	732
5.5.7	Validation .....	736
5.5.8	XML_Map.....	746
5.6	Text Data Processing transforms.....	766
5.6.1	Entity Extraction transform.....	766
<b>Chapter 6</b>	<b>Functions and Procedures.....</b>	<b>777</b>
6.1	About functions.....	777
6.1.1	Functions compared with transforms.....	777
6.1.2	Operation of a function.....	777
6.1.3	Arithmetic in date functions.....	778
6.1.4	Including functions in expressions.....	778
6.2	Built-in functions.....	780
6.2.1	Database and application functions.....	781
6.3	Descriptions of built-in functions.....	781
6.3.1	abs.....	790
6.3.2	add_months.....	791
6.3.3	ascii.....	792
6.3.4	avg.....	793
6.3.5	base64_decode.....	793
6.3.6	base64_encode.....	794
6.3.7	cast.....	795
6.3.8	chr.....	797
6.3.9	ceil.....	798
6.3.10	concat_date_time .....	799
6.3.11	count .....	799
6.3.12	count_distinct.....	800
6.3.13	current_configuration .....	801
6.3.14	current_system_configuration.....	802
6.3.15	dataflow_name .....	802
6.3.16	datastore_field_value.....	803
6.3.17	date_diff.....	803
6.3.18	date_part .....	804
6.3.19	day_in_month.....	806
6.3.20	day_in_week.....	807
6.3.21	day_in_year.....	808
6.3.22	db_type.....	809

6.3.23	db_version.....	810
6.3.24	db_database_name.....	812
6.3.25	db_owner.....	813
6.3.26	decode.....	813
6.3.27	decrypt_aes.....	815
6.3.28	decrypt_aes_ext.....	816
6.3.29	double_metaphone.....	817
6.3.30	encrypt_aes.....	818
6.3.31	encrypt_aes_ext.....	819
6.3.32	error_context.....	820
6.3.33	error_message.....	821
6.3.34	error_number.....	821
6.3.35	error_timestamp.....	821
6.3.36	exec.....	822
6.3.37	extract_from_xml.....	826
6.3.38	file_exists .....	828
6.3.39	fiscal_day.....	828
6.3.40	floor.....	829
6.3.41	gen_row_num_by_group.....	830
6.3.42	gen_row_num .....	831
6.3.43	gen_uuid.....	832
6.3.44	get_domain_description.....	832
6.3.45	get_env.....	833
6.3.46	get_error_filename .....	834
6.3.47	get_file_attribute.....	834
6.3.48	get_monitor_filename .....	835
6.3.49	get_trace_filename .....	836
6.3.50	greatest.....	837
6.3.51	host_name .....	838
6.3.52	ifthenelse.....	838
6.3.53	index.....	840
6.3.54	init_cap.....	841
6.3.55	interval_to_char.....	842
6.3.56	is_group_changed.....	843
6.3.57	is_set_env.....	844
6.3.58	is_valid_date.....	845
6.3.59	is_valid_datetime.....	846
6.3.60	is_valid_decimal.....	848
6.3.61	is_valid_double.....	849
6.3.62	is_valid_int.....	849
6.3.63	is_valid_real.....	850

6.3.64	is_valid_time.....	851
6.3.65	isempty .....	852
6.3.66	isweekend .....	853
6.3.67	job_name .....	854
6.3.68	Job_Run_ID.....	854
6.3.69	julian.....	855
6.3.70	julian_to_date.....	855
6.3.71	key_generation.....	856
6.3.72	last_date .....	857
6.3.73	least.....	858
6.3.74	length .....	859
6.3.75	literal.....	860
6.3.76	ln.....	862
6.3.77	load_to_xml.....	862
6.3.78	log.....	864
6.3.79	long_to_varchar.....	865
6.3.80	lookup .....	866
6.3.81	lookup_ext.....	869
6.3.82	lookup_seq.....	875
6.3.83	lower .....	878
6.3.84	lpad .....	879
6.3.85	lpad_ext.....	880
6.3.86	ltrim .....	881
6.3.87	ltrim_blanks .....	882
6.3.88	ltrim_blanks_ext .....	883
6.3.89	mail_to.....	884
6.3.90	match_pattern.....	886
6.3.91	match_regex .....	889
6.3.92	match_simple.....	896
6.3.93	max.....	897
6.3.94	min.....	898
6.3.95	mod.....	899
6.3.96	month.....	900
6.3.97	num_to_interval.....	900
6.3.98	nvl.....	902
6.3.99	power.....	902
6.3.100	previous_row_value.....	903
6.3.101	print .....	904
6.3.102	pushdown_sql .....	905
6.3.103	quarter.....	907
6.3.104	raise_exception .....	908



6.3.105	raise_exception_ext .....	908
6.3.106	rand.....	909
6.3.107	rand_ext.....	910
6.3.108	replace_substr .....	911
6.3.109	replace_substr_ext.....	912
6.3.110	repository_name .....	914
6.3.111	round.....	914
6.3.112	rpadd.....	915
6.3.113	rpadd_ext.....	916
6.3.114	rtrim .....	918
6.3.115	rtrim_blanks .....	919
6.3.116	rtrim_blanks_ext .....	920
6.3.117	sap_openhub_processchain_execute.....	920
6.3.118	sap_openhub_set_read_status.....	923
6.3.119	search_replace.....	925
6.3.120	set_cdc_checkpoint.....	929
6.3.121	set_env .....	930
6.3.122	sleep .....	931
6.3.123	soundex.....	931
6.3.124	sql .....	932
6.3.125	sqrt.....	934
6.3.126	smtp_to.....	935
6.3.127	substr.....	938
6.3.128	sum.....	940
6.3.129	sysdate.....	941
6.3.130	system_user_name .....	942
6.3.131	sys_time.....	942
6.3.132	table_attribute .....	943
6.3.133	to_char.....	944
6.3.134	to_date.....	947
6.3.135	to_decimal .....	948
6.3.136	to_decimal_ext .....	949
6.3.137	total_rows.....	950
6.3.138	trunc.....	951
6.3.139	truncate_table.....	952
6.3.140	upper.....	953
6.3.141	varchar_to_long.....	954
6.3.142	wait_for_file.....	955
6.3.143	week_in_month.....	956
6.3.144	week_in_year.....	957
6.3.145	WL_GetKeyValue .....	958

6.3.146	word.....	958
6.3.147	word_ext .....	959
6.3.148	workflow_name .....	961
6.3.149	year.....	962
6.4	Custom functions.....	962
6.4.1	To create a custom function.....	963
6.4.2	To edit an existing function.....	964
6.4.3	To replicate a custom function.....	965
6.4.4	To delete a custom function.....	965
6.5	About procedures.....	965
6.5.1	Overview.....	965
6.5.2	Requirements.....	966
6.5.3	Creating stored procedures in a database.....	967
6.5.4	Importing metadata for stored procedures .....	971
6.5.5	Structure of a stored procedure.....	972
6.5.6	Calling stored procedures.....	973
6.5.7	Checking execution status .....	977
<b>Chapter 7</b>	<b>Scripting Language.....</b>	<b>979</b>
7.1	To use the scripting language.....	979
7.2	Language syntax.....	979
7.2.1	Syntax for statements in scripts.....	979
7.2.2	Syntax for column and table references in expressions.....	980
7.2.3	Strings.....	981
7.2.4	Variables.....	982
7.2.5	Variable interpolation.....	983
7.2.6	Functions and stored procedures.....	983
7.2.7	Operators.....	984
7.2.8	NULL values.....	986
7.2.9	Debugging and Validation.....	988
7.2.10	Keywords.....	989
7.3	Sample scripts.....	992
7.3.1	Square function.....	992
7.3.2	RepeatString function.....	992
<b>Chapter 8</b>	<b>Metadata in Repository Tables and Views.....</b>	<b>995</b>
8.1	Auditing metadata.....	995
8.1.1	AL_AUDIT.....	995
8.1.2	AL_AUDIT_INFO.....	996
8.2	Imported metadata.....	997

8.2.1	AL_INDEX.....	997
8.2.2	AL_PCOLUMN.....	998
8.2.3	AL_PKEY.....	999
8.2.4	ALVW_COLUMNATTR.....	999
8.2.5	ALVW_COLUMNINFO.....	1000
8.2.6	ALVW_FKREL.....	1002
8.2.7	ALVW_MAPPING.....	1002
8.2.8	ALVW_TABLEATTR.....	1009
8.2.9	ALVW_TABLEINFO.....	1010
8.3	Internal metadata.....	1011
8.3.1	AL_LANG.....	1011
8.3.2	AL_LANGXMLTEXT.....	1012
8.3.3	AL_ATTR.....	1013
8.3.4	AL_SETOPTIONS.....	1014
8.3.5	AL_USAGE.....	1015
8.3.6	ALVW_FUNCINFO.....	1019
8.3.7	ALVW_PARENT_CHILD.....	1019
8.4	Operational metadata.....	1021
8.4.1	AL_HISTORY.....	1021
8.4.2	ALVW_FLOW_STAT.....	1022

## **Chapter 9 Data Quality repository statistics tables..... 1023**

9.1	Data Quality repository statistics tables.....	1023
9.2	Repository tables common columns.....	1025
9.3	Repository tables for USA and Global address cleanse.....	1026
9.3.1	ADDRINFOCODEDATA.....	1028
9.3.2	ADDRINFOCODESUMMARY.....	1029
9.3.3	ADDRSTATUSCODEDATA.....	1030
9.3.4	ADDRTYPESUMMARY.....	1031
9.3.5	ADDRVALIDATESUMMARY.....	1032
9.3.6	CSLSTATS.....	1034
9.3.7	DPVLACSLINKSUMMARY.....	1035
9.3.8	DSFAUGMENTSTATS.....	1037
9.3.9	DSFSEQUENCESTATS.....	1037
9.3.10	MEDSTATS.....	1038
9.3.11	NCOALCERTIFICATIONDATA.....	1039
9.3.12	NCOALINKSUMMARY.....	1044
9.3.13	PAFBALASTATS.....	1045
9.3.14	PSFORM3553DATA.....	1046
9.3.15	SENDRIGHTADDRACCURACY.....	1049
9.3.16	SERPADDRACCURACY.....	1051

9.3.17	USREGULATORYLOCKING.....	1052
9.4	Geocoder repository statistics tables.....	1053
9.4.1	GEO_ASSIGN_LEVEL.....	1054
9.4.2	GEO_INFO_CODE.....	1055
9.5	Match repository statistics tables.....	1055
9.5.1	MTBRACTION.....	1057
9.5.2	MTBRINFO.....	1058
9.5.3	MTBRKGRP.....	1059
9.5.4	MTBRKGRPINFO.....	1060
9.5.5	MTCMPCRIT.....	1061
9.5.6	MTCRITDEF.....	1062
9.5.7	MTDUPESDATA.....	1064
9.5.8	MTGSSRCBYSRCSTS.....	1064
9.5.9	MTGSSRCSTS.....	1065
9.5.10	MTINFO.....	1066
9.5.11	MTINSRCBYSRC.....	1066
9.5.12	MTINSRCGRPINFO.....	1067
9.5.13	MTINSRCINFO.....	1068
9.5.14	MTINSRCMSRC.....	1069
9.5.15	MTINSRCSELECT.....	1070
9.5.16	MTINSRCSTATS.....	1073
9.5.17	MTKEYDEF.....	1074
9.5.18	MTPROCESS.....	1075
9.5.19	MTRULESRES.....	1076
<b>Chapter 10</b>	<b>Locales and Multi-byte Functionality.....</b>	<b>1079</b>
10.1	Language packs.....	1079
10.1.1	To set locales in the Designer.....	1079
10.1.2	To set locales in the Locale Selector.....	1080
10.1.3	To set locales in UNIX or Linux.....	1080
10.1.4	Impact of locale settings on Data Services components.....	1081
10.2	Locale support.....	1081
10.2.1	Locale selection.....	1083
10.2.2	Code page support.....	1085
10.2.3	Guidelines for setting locales.....	1087
10.3	Multi-byte support.....	1092
10.3.1	Multi-byte string functions.....	1092
10.3.2	Numeric data types: assigning constant values.....	1092
10.3.3	Byte Order Mark characters.....	1093
10.3.4	Round-trip conversion.....	1094
10.3.5	Column sizing.....	1094

10.4	Limitations of multi-byte support.....	1095
10.5	Definitions.....	1095
10.6	Supported locales and encodings.....	1097
<b>Chapter 11</b>	<b>Python.....</b>	<b>1101</b>
11.1	Python.....	1101
11.1.1	About Python .....	1101
11.1.2	Create an expression with the Python Expression editor.....	1104
11.1.3	Built-in objects.....	1108
11.1.4	Defined classes and methods.....	1109
11.1.5	FIDataCollection class.....	1110
11.1.6	FIDataManager class.....	1114
11.1.7	FIDataRecord class.....	1116
11.1.8	FIProperties class.....	1118
11.1.9	FIPythonString class.....	1119
11.1.10	Python examples.....	1121
<b>Chapter 12</b>	<b>Hadoop.....</b>	<b>1127</b>
12.1	Prerequisites.....	1127
12.1.1	Configuring Hadoop for text data processing.....	1128
12.2	Connecting to HDFS.....	1128
12.3	Connecting to Hive.....	1129
12.3.1	Adding, configuring, and starting a Hive adapter instance .....	1129
12.3.2	Adding and configuring a Hive adapter datastore.....	1130
12.3.3	About partitions.....	1130
<b>Chapter 13</b>	<b>Reserved Words.....</b>	<b>1133</b>
13.1	About Reserved Words.....	1133
<b>Chapter 14</b>	<b>Glossary.....</b>	<b>1137</b>
<b>Index</b>		<b>1155</b>



# Introduction

## 1.1 Welcome to SAP BusinessObjects Data Services

### 1.1.1 Welcome

SAP BusinessObjects 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 BusinessObjects Data Services, IT organizations can maximize operational efficiency with a single solution to improve data quality and gain access to heterogeneous sources and applications.

### 1.1.2 Documentation set for SAP BusinessObjects Data Services

You should become familiar with all the pieces of documentation that relate to your SAP BusinessObjects Data Services product.

Document	What this document provides
<i>Administrator's Guide</i>	Information about administrative tasks such as monitoring, lifecycle management, security, and so on.
<i>Customer Issues Fixed</i>	Information about customer issues fixed in this release.
<i>Designer Guide</i>	Information about how to use SAP BusinessObjects Data Services Designer.
<i>Documentation Map</i>	Information about available SAP BusinessObjects Data Services books, languages, and locations.

Document	What this document provides
<i>Installation Guide for Windows</i>	Information about and procedures for installing SAP BusinessObjects Data Services in a Windows environment.
<i>Installation Guide for UNIX</i>	Information about and procedures for installing SAP BusinessObjects Data Services in a UNIX environment.
<i>Integrator's Guide</i>	Information for third-party developers to access SAP BusinessObjects Data Services functionality using web services and APIs.
<i>Master Guide</i>	Information about the application, its components and scenarios for planning and designing your system landscape. Information about SAP BusinessObjects Information Steward is also provided in this guide.
<i>Management Console Guide</i>	Information about how to use SAP BusinessObjects Data Services Administrator and SAP BusinessObjects Data Services Metadata Reports.
<i>Performance Optimization Guide</i>	Information about how to improve the performance of SAP BusinessObjects Data Services.
<i>Reference Guide</i>	Detailed reference material for SAP BusinessObjects Data Services Designer.
<i>Release Notes</i>	Important information you need before installing and deploying this version of SAP BusinessObjects Data Services.
<i>Technical Manuals</i>	<p>A compiled “master” PDF of core SAP BusinessObjects Data Services books containing a searchable master table of contents and index:</p> <ul style="list-style-type: none"> <li>• <i>Administrator's Guide</i></li> <li>• <i>Designer Guide</i></li> <li>• <i>Reference Guide</i></li> <li>• <i>Management Console Guide</i></li> <li>• <i>Performance Optimization Guide</i></li> <li>• <i>Supplement for J.D. Edwards</i></li> <li>• <i>Supplement for Oracle Applications</i></li> <li>• <i>Supplement for PeopleSoft</i></li> <li>• <i>Supplement for Salesforce.com</i></li> <li>• <i>Supplement for Siebel</i></li> <li>• <i>Supplement for SAP</i></li> <li>• <i>Workbench Guide</i></li> </ul>
<i>Text Data Processing Extraction Customization Guide</i>	Information about building dictionaries and extraction rules to create your own extraction patterns to use with Text Data Processing transforms.
<i>Text Data Processing Language Reference Guide</i>	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.
<i>Tutorial</i>	A step-by-step introduction to using SAP BusinessObjects Data Services.



Document	What this document provides
<i>Upgrade Guide</i>	Release-specific product behavior changes from earlier versions of SAP BusinessObjects Data Services to the latest release. This manual also contains information about how to migrate from SAP BusinessObjects Data Quality Management to SAP BusinessObjects Data Services.
<i>What's New</i>	Highlights of new key features in this SAP BusinessObjects Data Services release. This document is not updated for support package or patch releases.
<i>Workbench Guide</i>	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
<i>Supplement for J.D. Edwards</i>	Information about interfaces between SAP BusinessObjects Data Services and J.D. Edwards World and J.D. Edwards OneWorld.
<i>Supplement for Oracle Applications</i>	Information about the interface between SAP BusinessObjects Data Services and Oracle Applications.
<i>Supplement for PeopleSoft</i>	Information about interfaces between SAP BusinessObjects Data Services and PeopleSoft.
<i>Supplement for Salesforce.com</i>	Information about how to install, configure, and use the SAP BusinessObjects Data Services Salesforce.com Adapter Interface.
<i>Supplement for SAP</i>	Information about interfaces between SAP BusinessObjects Data Services, SAP Applications, SAP Master Data Services, and SAP NetWeaver BW.
<i>Supplement for Siebel</i>	Information about the interface between SAP BusinessObjects Data Services and Siebel.

We also include these manuals for information about SAP BusinessObjects Information platform services.

Document	What this document provides
<i>Information Platform Services Administrator's Guide</i>	Information for administrators who are responsible for configuring, managing, and maintaining an Information platform services installation.
<i>Information Platform Services Installation Guide for UNIX</i>	Installation procedures for SAP BusinessObjects Information platform services on a UNIX environment.
<i>Information Platform Services Installation Guide for Windows</i>	Installation procedures for SAP BusinessObjects Information platform services on a Windows environment.

## 1.1.3 Accessing documentation

You can access the complete documentation set for SAP BusinessObjects Data Services in several places.

### 1.1.3.1 Accessing documentation on Windows

After you install SAP BusinessObjects Data Services, you can access the documentation from the Start menu.

1. Choose **Start > Programs > SAP BusinessObjects Data Services 4.1 > Data Services Documentation > All Guides**.
2. Click the appropriate shortcut for the document that you want to view.

### 1.1.3.2 Accessing documentation on UNIX

After you install SAP BusinessObjects 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.

### 1.1.3.3 Accessing documentation from the Web

You can access the complete documentation set for SAP BusinessObjects Data Services from the SAP BusinessObjects 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.

## 1.1.4 SAP BusinessObjects information resources

A global network of SAP BusinessObjects 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 <a href="http://service.sap.com/">http://service.sap.com/</a>	Information about SAP Business User Support programs, as well as links to technical articles, downloads, and online forums. Consulting services can provide you with information about how SAP BusinessObjects 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 BusinessObjects can offer a training package to suit your learning needs and preferred learning style.
Product documentation <a href="http://help.sap.com/bods/">http://help.sap.com/bods/</a>	SAP BusinessObjects product documentation.
Supported Platforms (Product Availability Matrix) <a href="https://service.sap.com/PAM">https://service.sap.com/PAM</a>	Get information about supported platforms for SAP BusinessObjects Data Services.  Use the search function to search for Data Services. Click the link for the version of Data Services you are searching for.

## 1.2 Overview of this guide

### 1.2.1 About this guide

The *Data Services Reference Guide* provides a 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.

## **1.2.2 Who should read this guide**

This and other SAP BusinessObjects 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.

# 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 Topics**

- [Characteristics of objects](#)
- [Descriptions of objects](#)

## 2.1 Characteristics of objects

This section discusses common characteristics of all the objects.

**Related Topics**

- [Object classes](#)
- [Object options, properties, and attributes](#)

### 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

### 2.1.1.1 Reusable objects

After you define and save a reusable object, SAP BusinessObjects 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.

### 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.

## 2.1.2 Object options, properties, and attributes

Each object is associated with a set of options, properties, and attributes:

- *Options* control the operation of an object. For example, in a datastore, an option is the name of the database to which the datastore connects.
- *Properties* document 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* provide 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.

## 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 an error 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.



Object	Class	Description
Excel workbook format	Reusable	Defines the format for an Excel workbook source.
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.

Object	Class	Description
Table	Reusable	<p>Indicates an external DBMS table for which metadata has been imported, or the target table into which data is or has been placed.</p> <p>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.</p>
Target	Single-use	An object in which the software loads extracted and transformed data in a data flow.
Template table	Reusable	<p>A new table you want added to a database.</p> <p>All datastores except SAP datastores have a default template that you can use to create any number of tables in the datastore.</p> <p>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.</p>
Transform	Reusable	<p>Performs operations on data sets.</p> <p>Requires zero or more data sets; produces zero or one data set (which may be split).</p>
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.

Object	Class	Description
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.

## 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 Topics

- [Designer Guide: Creating annotations](#)

## 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 BusinessObjects 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 Topics

- [Designer Guide: Setting global variable values](#)

### 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	<p>Select this check box if you want to display cache statistics in the Performance Monitor in the Administrator. (The default is cleared.)</p> <p><b>Note:</b> Use this option only if you want to look at the cache size.</p> <p>For more information, see the Using Caches section in the <i>Performance Optimization Guide</i>.</p>
Collect statistics for optimization	<p>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.</p> <p>For more information, see Caching sources in the <i>Performance Optimization Guide</i>.</p> <p>For more information, see the Using Caches section in the <i>Performance Optimization Guide</i>.</p>
Disable data validation statistics collection	<p>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.)</p> <p>For more information about data validation statistics, see Data Validation dashboards Settings control panel" in the <i>Management Console Guide</i>.</p> <p>For more information about data validation statistics, see the Data Validation Dashboard Reports section in the <i>Management Console Guide</i>.</p>

Options	Description
Distribution level	<p>Select the level within a job that you want to distribute to multiple job servers for processing:</p> <ul style="list-style-type: none"> <li>• Job - The whole job will execute on an available Job Server.</li> <li>• Data flow - Each data flow within the job can execute on an available Job Server.</li> <li>• 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.</li> </ul> <p>For more information, see Using grid computing to distribute data flows execution in the <i>Performance Optimization Guide</i>.</p> <p>For more information, see the Distributing Data Flow Execution section in the <i>Performance Optimization Guide</i>.</p>
Enable auditing	<p>Clear this check box if you do not want to collect audit statistics for this specific job execution. (The default is selected.)</p> <p>For more information about auditing, see Using Auditing in the <i>Designer Guide</i>.</p> <p>For more information about auditing, see the Data Assessment section in the <i>Designer Guide</i>.</p>
Enable recovery	<p>(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.</p> <p>See Automatically recovering jobs in the <i>Designer Guide</i> for information about the recovery options.</p> <p>See the Recovery Mechanisms section in the <i>Designer Guide</i> for information about the recovery options.</p> <p>This property is only available as a run-time property. It is not available as a default property.</p>
Export Data Quality reports	<p>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 <code>&lt;DS_COMMON_DIR&gt;\DataQuality\reports\repository\job</code>.</p>

Options	Description
Job Server or Server group	<p>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.</p> <p>For an introduction to server groups, see Server group architecture in the <i>Management Console Guide</i>.</p> <p>For an introduction to server groups, see the Server Groups section in the <i>Management Console Guide</i>.</p> <p>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:</p> <ul style="list-style-type: none"> <li>• Directory and file names for source and target files</li> <li>• Bulk load directories</li> </ul>
Monitor sample rate (# of seconds)	<p>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.</p> <p>For example, if you enter 30, the logs update every 30 seconds.</p> <p>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.</p> <p>Setting the rate to 0 disables the monitoring feature.</p> <p><b>Note:</b></p> <p>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.</p>
Print all trace messages	<p>Select this check box to print all trace messages to the trace log file for the current Job Server.</p> <p>Selecting this option overrides the trace properties set on the <b>Trace</b> tab.</p>
Recover from last failed execution	<p>(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.</p> <p>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.</p>



Options	Description
System configuration	<p>Select the system configuration to use when executing this job. A system configuration defines a set of datastore configurations, which define the datastore connections.</p> <p>For more information, see <i>Creating and managing multiple datastore configurations</i> in the <i>Designer Guide</i>.</p> <p>For more information, see the <i>Datastores</i> section in the <i>Designer Guide</i>.</p> <p>If a system configuration is not specified, the software uses the default datastore configuration for each datastore.</p> <p>This option is a run-time property. This option is only available if there are system configurations defined in the repository.</p>
Use collected statistics	<p>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.)</p> <p>For more information, see <i>Monitoring and tuning caches</i> in the <i>Performance Optimization Guide</i>.</p> <p>For more information, see section 6: <i>Using Caches</i> in the <i>Performance Optimization Guide</i>.</p>

### 2.2.2.2 Trace properties

Use trace properties to select the information that SAP BusinessObjects 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	<p>Writes messages exchanged between the Access Server and a service provider, including:</p> <ul style="list-style-type: none"> <li>• The registration message, which tells the Access Server that the service provider is ready</li> <li>• The request the Access Server sends to the service to execute</li> <li>• The response from the service to the Access Server</li> <li>• Any request from the Access Server to shut down</li> </ul>
Assemblers	<p>Writes messages for Substitution Parameter and SDK Transforms:</p> <ul style="list-style-type: none"> <li>• Substitution parameters - Writes trace messages such as substitution configuration used and the actual value substituted for each substitution parameter.</li> <li>• 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.</li> </ul>
Audit Data	<p>Writes a message when auditing:</p> <ul style="list-style-type: none"> <li>• Collects a statistic at an audit point</li> <li>• Determines if an audit rule passes or fails</li> </ul>
Data Flow	<p>Writes a message when the data flow starts, when the data flow successfully finishes, or when the data flow terminates due to error.</p> <p>This trace also reports when the bulk loader starts, any bulk loader warnings occur, and when the bulk loader successfully completes.</p>
IDoc file reader	<p>Writes a message when reading IDoc files including:</p> <ul style="list-style-type: none"> <li>• start reading the IDoc file</li> <li>• stop reading the IDoc file</li> <li>• result of the IDoc file validation</li> </ul>
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 Dataflow	For consulting and customer assurance use.

Trace	Description
RFC Function	Writes a message related to RFC calls including: <ul style="list-style-type: none"><li>• start of RFC call</li><li>• end of RFC call</li><li>• a message for each record received from the software for the RFC call</li></ul>
Row	Writes a message when a transform imports or exports a row.
SAP Table Reader	Writes messages from readers of SAP system tables including: <ul style="list-style-type: none"><li>• start reading from table</li><li>• stop reading from table</li><li>• start of connection to SAP system where table is present</li><li>• end of connection to SAP system</li></ul>
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: <ul style="list-style-type: none"><li>• 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.</li><li>• A function is called by the script.</li><li>• The script successfully completes.</li></ul>
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: <ul style="list-style-type: none"><li>• Every row retrieved by the named query before the SQL is submitted in the key_generation function</li><li>• 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).</li><li>• When mail is sent using the mail_to function.</li></ul>

Trace	Description
SQL Loaders	<p>Writes a message when the bulk loader:</p> <ul style="list-style-type: none"> <li>• Starts</li> <li>• Submits a warning message</li> <li>• Completes successfully</li> <li>• Completes unsuccessfully, if the <b>Clean up bulk loader directory after load</b> option is selected</li> </ul> <p>Additionally, for Microsoft SQL Server and Sybase ASE, writes when the SQL Server bulk loader:</p> <ul style="list-style-type: none"> <li>• Completes a successful row submission</li> <li>• Encounters an error</li> </ul> <p>This instance reports all SQL that the software submits to the target database, including:</p> <ul style="list-style-type: none"> <li>• When a <code>truncate table</code> command executes if the <b>Delete data from table before loading</b> option is selected.</li> <li>• Any parameters included in PRE-LOAD SQL commands</li> <li>• Before a batch of SQL statements is submitted</li> <li>• When a template table is created (and also dropped, if the <b>Drop/Create</b> option is turned on)</li> <li>• When a <code>delete from table</code> command executes if auto correct is turned on (Informix environment only).</li> </ul> <p>This trace also writes all rows that the software loads into the target.</p>
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	<p>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.</p> <p>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).</p>

Trace	Description
Stored Procedure	Writes a message when the software invokes a stored procedure. This trace reports: <ul style="list-style-type: none"> <li>• When the stored procedure starts</li> <li>• The SQL query submitted for the stored procedure call</li> <li>• The value (or values) of the input parameter (or parameters)</li> <li>• The value (or values) of the output parameter (or parameters)</li> <li>• The return value (if the stored procedure is a stored function)</li> <li>• When the stored procedure finishes</li> </ul>
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 dataflow 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.

### 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 workflow, 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

Exception group	Group number	Description
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

### 2.2.3.1 Catch error functions

The following table describes error functions that you can use in the script that your catch work flow executes.

**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.
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 Topics

- [Designer Guide: Work flows, Example: Catching details of an error](#)

### 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

The syntax for a try/catch block within a script is as follows:

```
try
begin
  steps
end
catch(integer_constants)
begin
  steps
end
```

Where:

<i>steps</i>	Catch error functions, other function calls, if statements, or other statements you want to perform for an error in the specified exception group number.
<i>integer_constants</i>	<p>One or more exception group numbers that you want to catch.</p> <p>Use a comma to separate exception group numbers. For example,</p> <pre>catch(1002,1003)</pre> <p>Specify ALL to catch all exceptions:</p> <pre>catch(ALL)</pre>

#### Related Topics

- [Designer Guide: Work flows, Example: Catching details of an error](#)



## 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.

### 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.

Format option	Description
Expand OCCURS	<p>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:</p> <ul style="list-style-type: none"> <li>Getting a sequential suffix for each repetition: fieldname_1, fieldname_2, etc. (expanded view)</li> <li>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.</li> </ul> <p>If a copybook contains more than one OCCURS group, you must select this box. The default is selected.</p>
Ignore redefines	Determines whether or not to ignore REDEFINES clauses.
Source format	<p>The format of the copybook source code. Options include:</p> <ul style="list-style-type: none"> <li><b>Free</b>—All characters on the line can contain COBOL source code.</li> <li><b>Smart mode</b>—the software attempts 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.</li> <li><b>Standard</b>—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).</li> </ul>
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	<p>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 <b>Name</b> field. During design, you can specify a file in one of the following ways:</p> <ul style="list-style-type: none"> <li>For a file located on the computer where the Designer runs, you can use the Browse button.</li> <li>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.</li> </ul>
File name	<p>Type the name or browse to the COBOL copybook data file. You can use variables or wild cards (* or ?).</p> <p>If you leave <b>Directory</b> blank, then type the full path and file name here.</p>
Type	<p>Specifies the record format—fixed or variable:</p> <ul style="list-style-type: none"> <li><b>Fixed(F)</b></li> <li><b>Variable(V)</b></li> </ul>
Has record length	<p>Specifies whether variable-length records of the data file contain information about the length of each record. The possible values are:</p> <ul style="list-style-type: none"> <li><b>2-byte</b> integer</li> <li><b>2-byte followed by 0x0000</b> (integer followed by two zero bytes)</li> <li><b>4-byte</b> integer</li> <li><b>None</b>—No length information at the beginning of each record</li> </ul>
Record size	<p>Defines fixed record length in bytes. All records in the file have this length (padded, if necessary).</p>
Record trailer length	<p>Specifies the length of extra character padding in bytes at the end of each record.</p>
Has record mark	<p>Defines whether there is an extra byte in the beginning of each record's data.</p>
Integer format	<p>Describes how the existing data file stores binary data:</p> <ul style="list-style-type: none"> <li><b>Big endian</b>—the most significant byte comes first</li> <li><b>Little endian</b>—the least significant byte comes first</li> </ul>
Code page	<p>Specifies the character encoding of character data in the data file.</p>
Skip first	<p>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.</p>
Read total	<p>Defines the number of records to read and process. For added flexibility, you can enter a variable for this option.</p>

Data file option	Description
Low Value and High Value	<p>Identifies a valid hexadecimal value for a low value, high value, or both from the copybook and applies the associated <b>Action</b>. 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.</p> <p>For example, if the source field is binary 0x40, enter a Low Value of 0x40 and select the action <b>Convert to NULL</b>. The result would be as follows for these data types:</p> <ul style="list-style-type: none"> <li>• Char—Character represented by 0x40</li> <li>• Packed decimal—NULL</li> <li>• Binary—0x40</li> </ul>
Action	<p>For a Low or High Value, applies one of the following actions:</p> <ul style="list-style-type: none"> <li>• <b>No conversion</b>—Reads the value as an ASCII character (default).</li> <li>• <b>Convert to NULL</b>—Converts the given value to NULL.</li> <li>• <b>Convert to 0</b>—Converts the given value to 0.</li> </ul>

### 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.
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 <b>Name</b> field.
File name	<p>Type or browse to the COBOL copybook data file <b>Name</b>. You can use variables or wild cards (* or ?).</p> <p>If you leave <b>Directory</b> blank, then type a full path and file name here.</p>
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.

Data access option	Description
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	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.

## 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	<p>Makes the source table an embedded data flow port.</p> <p>For more information, see the Embedded Data Flows section in the <i>Designer Guide</i>.</p>

Table 2-15: Performance

Source option	Description
Join rank	<p>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.</p> <p>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.</p> <p>Must be a non-negative integer. Default value is 0.</p> <p>For more information, see the Other Tuning Techniques section in the <i>Performance Optimization Guide</i>.</p>
Cache	<p>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.</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• <b>Yes:</b> The source is always cached unless it is the outer-most source in a join.</li> <li>• <b>No:</b> The source is never cached.</li> </ul> <p>The default is <b>Yes</b>.</p> <p>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.</p>

Table 2-16: Error handling

Source option	Description
Log data conversion warnings	Determines whether to include data-type conversion warnings in the error log. Defaults to <b>Yes</b> .
Maximum warnings to log	If Log data conversion warnings is enabled, you can limit how many warnings are logged. Defaults to <b>{no limit}</b> .

Table 2-17: Include file name column

Source option	Description
Include file name column	<p>Determines whether to add a column that contains the source file name in the source output. Defaults to <b>No</b>.</p> <p>Change the value to <b>Yes</b> when you want to identify the source file in situations such as the following:</p> <ul style="list-style-type: none"> <li>You specified a wildcard character to read multiple source COBOL copybooks at one time</li> <li>You load from different source copybooks on different runs</li> </ul>
Modify	If the file name is included, this button enables you to modify <b>File name column</b> and <b>Column size</b> .
File name column	If the file name is included, the name of the column that holds the source file name. Defaults to <b>DI_FILENAME</b> .
Column size	<p>If the file name is included, the size (in characters) of the column that holds the source file name.</p> <p>Defaults to <b>100</b>. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.</p>
Include path	If the file name is included, determines whether to include the full path name of the source file. Defaults to <b>No</b> .

### 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 <i>Reference Guide</i> .

Field clauses option	Description
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 Topics

- [Import or Edit COBOL copybook format options](#)

## 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 Topics**

- [Smart editor](#)

## 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 BusinessObjects 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 Topics

- [Performance Optimization Guide: Viewing SQL](#)

## 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 BusinessObjects 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.

### 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 Topics

- [Performance Optimization Guide: Degree of parallelism](#)

### 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 Topics**

- [Performance Optimization Guide: Caching sources](#)

## 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 BusinessObjects 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

### 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 BusinessObjects 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.

#### **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 a 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.

Button	Button name	Description
	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.
	Sort Configurations (Ascending)	Arranges the configurations by their names in ascending order. The arrangement is sensitive to the computer's system locale.
	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.
	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.

Button	Button name	Description
	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
Dataflow	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 BusinessObjects 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 Topics

- [Performance Optimization Guide: Maximizing Push-Down Operations, DataTransfer transform for push-down operations](#)
- [Working with Aliases](#)
- [Designer Guide: Datastores, Adapter datastores](#)
- [Designer Guide: Datastores, Linked datastores](#)

#### 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.

#### 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.

#### 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**.

#### 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 BusinessObjects 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.

#### 2.2.7.2 Database datastores

You can define datastores so that SAP BusinessObjects Data Services can read from and write to the following types of databases:

- Attunity Connector (use for mainframe systems)
- Data Federator (read only)
- DB2
- SAP HANA (ODBC)
- HP Neoview
- Informix
- Memory
- Microsoft SQL Server
- MySQL
- Netezza
- ODBC
- Oracle
- Persistent Cache
- Sybase ASE

- 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.

**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 Topics**

- [ODBC](#)
- [Designer Guide: What are datastores](#)
- [Performance Optimization Guide: Using Bulk Loading](#)

### 2.2.7.2.1 Attunity

Table 2-23: 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 BusinessObjects 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 2-24: 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 <b>Browse</b>	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.

Table 2-25: 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 2-26: 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 2-27: Aliases

Attunity option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

### 2.2.7.2.2 Data Federator

Any decimal column imported to SAP BusinessObjects Data Services from an SAP BusinessObjects Data Federator data source is converted to the decimal precision and scale(28,6).

Any varchar column imported to the software from a SAP BusinessObjects 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 BusinessObjects Data Federator data source.

### 2.2.7.2.3 DB2

Table 2-28: 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 <b>Database server name</b> , <b>Database name</b> , and <b>Port</b> .  If you select this checkbox, you must fill in <b>Data source name</b>
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 <b>Use data source name (DSN)</b> .
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 <b>Use data source name (DSN)</b> .
Port	Five digit integer Default: 50000	Enter the number of the database port.  This option is required if you did not select <b>Use data source name (DSN)</b> .

DB2 option	Possible values	Description
Data source name	Refer to the requirements of your database	<p>Type the data source name defined in DB2 for connecting to your database.</p> <p>This option is required when you select <b>Use data source name (DSN)</b>.</p> <p>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.</p>
User name	Alphanumeric characters and underscores	Enter the user name of the account through which SAP BusinessObjects Data Services accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.

Table 2-29: 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 datastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click <b>Browse</b>	<p>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.</p> <p>You can enter a variable for this option.</p>
Overflow file directory	Directory path or click <b>Browse</b>	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.

Table 2-30: Locale

DB2 option	Possible values	Description
Language	-	See the section "Locales and Multi-Byte Functionality."



DB2 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 2-31: 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 <b>Browse</b>	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 2-32: 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 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.
Session		

DB2 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.
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

#### 2.2.7.2.4 HP Neoview

Table 2-33: 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		<p>The Enable automatic data transfer check box is selected by default when you create a new datastore and you chose Database for Datastore type.</p> <p>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.</p>

Table 2-34: 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 <b>Browse</b>	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 <b>Browse</b>	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.

Table 2-35: 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 2-36: 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 2-37: 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.

### 2.2.7.2.5 Informix

Table 2-38: 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 <b>Database server name</b> , <b>Database name</b> , and <b>Port</b> .  If you select this checkbox, you must fill in <b>Data source name</b> .
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 <b>Use data source name (DSN)</b> .

Informix option	Possible values	Description
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 <b>Use data source name (DSN)</b> .
Port	Four digit integer Default: 1526	Type the port number to connect to this database. This option is required if you did not select <b>Use data source name (DSN)</b> .
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 <b>Use data source name (DSN)</b> .
User name	Alphanumeric characters and underscores	Enter the user name of the account through which SAP BusinessObjects Data Services accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.

Table 2-39: 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 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 <b>Browse</b>	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 <b>Browse</b>	Enter the location of overflow files written by target tables in this datastore.

Table 2-40: 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 2-41: 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.

**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.

**2.2.7.2.6 Memory**

Table 2-42: 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 Topics

- [Locales and Multi-byte Functionality](#)

### 2.2.7.2.7 Microsoft SQL Server

#### Note:

In order to use Microsoft SQL Server as a source or target datastore when SAP BusinessObjects Data Services is running on a UNIX platform, you must use an ODBC driver, such as the DataDirect ODBC driver.

For more information about how to obtain the driver, see the Platforms Availability Report (PAR) available in the **SAP BusinessObjects Support > Documentation > Supported Platforms** section of the SAP Service Marketplace: <http://service.sap.com/bosap-support>

Table 2-43: 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 SAP BusinessObjects 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.
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.

Table 2-44: 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 <b>Browse</b>	Enter the location of overflow files written by target tables in this datastore. You can enter a variable for this option.

Table 2-45: 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 2-46: 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 2-47: 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 2-48: 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

### 2.2.7.2.8 MySQL

Table 2-49: 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 <b>Database server name</b> , <b>Database name</b> , and <b>Port</b> .  If you select this checkbox, you must fill in <b>Data source name</b>
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 <b>Use data source name (DSN)</b> .
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 <b>Use data source name (DSN)</b> .
Port	Refer to the requirements of your database	Enter the number of the database port.  This option is required if you did not select <b>Use data source name (DSN)</b> .

MySQL option	Possible values	Description
Data source name	Refer to the requirements of your database	<p>Type the data source name defined in MySQL for connecting to your database.</p> <p>This option is required when you select <b>Use data source name (DSN)</b>.</p> <p>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.</p>
User name	Alphanumeric characters and underscores	Enter the user name of the account through which SAP BusinessObjects Data Services accesses the database.
Password	Alphanumeric characters, underscores, and punctuation	Enter the user's password.
Enable automatic data transfer		<p>The Enable automatic data transfer check box is selected by default when you create a new datastore and you chose Database for Datastore type.</p> <p>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.</p>

Table 2-50: Connection

MySQL option	Possible values	Description
Additional connection parameters	Alphanumeric characters and underscores, or blank	<p>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:</p> <pre>&lt;parameter1=value1; parameter2=value2&gt;</pre>

Table 2-51: 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 <b>Browse</b>	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 <b>Browse</b>	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 <b>Browse</b>	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 2-52: Locale

MySQL 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 2-53: 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 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 2-54: 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.

MySQL option	Values	Description
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 2-55: 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 2-56: 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.

### 2.2.7.2.9 Netezza

Table 2-57: 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	<p>Select to use DSN to connect to the database.</p> <p>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 <b>Database server name</b>, <b>Database name</b>, and <b>Port</b>.</p> <p>If you select this checkbox, you must fill in <b>Data source name</b></p>

Netezza option	Possible values	Description
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 <b>Use data source name (DSN)</b> .
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 <b>Use data source name (DSN)</b> .
Port	Refer to the requirements of your database	Enter the number of the database port. This option is required if you did not select <b>Use data source name (DSN)</b> .
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 <b>Use data source name (DSN)</b> . 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 BusinessObjects 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 data-store 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 2-58: Connection

Netezza option	Possible values	Description
Additional connection parameters	Alphanumeric characters and underscores, or blank	<p>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:</p> <pre>&lt;parameter1=value1; parameter2=value2&gt;</pre>

Table 2-59: 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 <b>Browse</b>	<p>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.</p> <p>You can enter a variable for this option.</p>
Overflow file directory	Directory path or click <b>Browse</b>	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 <b>Browse</b>	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 2-60: 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."

Netezza option	Possible values	Description
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 2-61: 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 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 2-62: 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).
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).



Netezza option	Possible values	Description
Decimal separator	.(period) 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 2-63: 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 2-64: 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.

### 2.2.7.2.10 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.

Option	Possible values	Description
Use data source name (DSN)	Check box selected or not selected	<p>Select to use DSN to connect to the database.</p> <p>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 <b>Database server name</b> and <b>Port</b>.</p> <p>If you select this checkbox, you must fill in <b>Data source name</b>.</p>
Database server name	Computer name	<p>Enter the name of the computer where the SAP HANA server is located.</p> <p>This option is required if you did not select <b>Use data source name (DSN)</b>.</p>
Port	<p>Five digit integer</p> <p>Default: 30015</p>	<p>Enter the port number to connect to this SAP HANA Server.</p> <p>This option is required if you did not select <b>Use data source name (DSN)</b>.</p>
Data source name	Refer to the requirements of your database	<p>Select or type the Data Source Name defined in the ODBC Administrator for connecting to your database.</p> <p>This option is required when you select <b>Use data source name (DSN)</b>.</p>
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		<p>The Enable automatic data transfer check box is selected by default when you create a new datastore and you chose Database for Datastore type.</p> <p>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.</p>

Option	Possible 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>
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.

### 2.2.7.2.11 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 BusinessObjects 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 2-66: 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.

ODBC option	Values	Description
User name	Alphanumeric characters and underscores	Enter the user name of the account through which the software accesses the database. <b>Note:</b> If you use the Neoview utility Nvtencsrv 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. <b>Note:</b> If you use the Neoview utility Nvtencsrv 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 2-67: 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 2-68: 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 <b>Browse</b>	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.

ODBC option	Values	Description
Overflow file directory	Directory path or click <b>Browse</b>	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 <b>Browse</b>	A working directory on the database server that stores files such as logs. Must be defined to use FTP.

Table 2-69: 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 2-70: 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 2-71: 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	<p>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.</p> <p><b>Note:</b> Configure the FTP server to accept an absolute path.</p>

Table 2-72: ODBC Capability Support

ODBC option	Values	Description
Array fetch	Automatic, No	If you encounter errors when reading from an ODBC data store, 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 <b>No</b> value to turn off the array fetch capability. The software fetches one row at a time from the ODBC data source. The <b>No</b> value causes all Source Table Editors and SQL Transform Editors that use this ODBC data store to not display the <b>Array fetch size</b> 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 2-73: 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 2-74: 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.



ODBC option	Values	Description
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 2-75: 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 2-76: 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.

ODBC option	Values	Description
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 2-77: 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).
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.

ODBC option	Values	Description
NVARCHAR type name	<Unknown> NVARCHAR NVARCHAR2	<p>For loading multibyte data to template tables, select the option depending on the database type:</p> <p><b>NVARCHAR:</b> All supported databases except Oracle</p> <p><b>NVARCHAR2:</b> Oracle databases only</p> <p>In the template table target editor, setting the option <b>Use NVARCHAR for VARCHAR columns in supported databases</b> to Yes enables this data type conversion. See also <a href="#">Template table</a>.</p>

Table 2-78: 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 2-79: Aliases

ODBC option	Values	Description
Aliases		Enter the alias name and the owner name to which the alias name maps.

### Related Topics

- [Administrator's Guide: Configuring ODBC data sources on UNIX](#)

### 2.2.7.2.12 Oracle

Table 2-80: 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.

Oracle option	Possible values	Description
Use TNS name	Check box selected or not selected	<p>Select to use TNS to connect to the database.</p> <p>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 <b>Hostname</b>, <b>SID</b>, and <b>Port</b>.</p> <p>If you select this checkbox, you must fill in <b>TNS name</b></p>
Hostname	Computer name, fully qualified domain name, or IP address	<p>Enter the name of machine where the Oracle Server instance is located.</p> <p>This option is required if you did not select <b>Use TNS name</b>.</p>
SID	Refer to the requirements of your database	<p>Enter the System ID for the Oracle database.</p> <p>This option is required if you did not select <b>Use TNS name</b>.</p>
Port	Four digit integer Default: 1521	<p>Enter the port number to connect to this Oracle Server.</p> <p>This option is required if you did not select <b>Use TNS name</b>.</p>
TNS name	Refer to the requirements of your database	<p>Enter an existing Oracle Transparent Network Substrate (TNS) name through which the software accesses sources and targets defined in this datastore.</p> <p>This option is required when you select <b>Use TNS name</b>.</p>
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 2-81: 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 commit size for target tables in this datastore. You can overwrite this value for individual target tables.
Bulk loader directory	Directory path or click <b>Browse</b>	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 <b>Browse</b>	Enter the location of overflow files written by target tables in this datastore. You can enter a variable for this option.

Table 2-82: 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 2-83: 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 2-84: Oracle Miscellaneous

Oracle option	Possible values	Description
Default precision for number	1 <= precision <= 96	Enter the total number of digits in the value.

Oracle option	Possible values	Description
Default scale for number	0 <= scale <= precision	Enter the number of digits to the right of the decimal point.

Table 2-85: 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 2-86: 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

### 2.2.7.2.13 Persistent Cache

Table 2-87: Locale

Persistent Cache option	Possible values	Description
Server code page	-	See the section "Locales and Multi-Byte Functionality."

Table 2-88: 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)

### 2.2.7.2.14 Sybase ASE

Table 2-89: Main window

Sybase ASE option	Possible values	Description
Database version	Sybase ASE <version number>	Select the version of your Sybase ASE client. This is the version of Sybase that this datastore accesses.
Database server name	Computer name	<p>Enter the name of the computer where the Sybase ASE instance is located.</p> <p><b>Note:</b> For UNIX Job Servers, when logging in to a 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.</p>
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 2-90: General

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 <b>Browse</b>	Enter the location of overflow files written by target tables in this datastore. A variable can also be used.

Table 2-91: Locale

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 2-92: Session

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 delimited by semicolon.

Table 2-93: Aliases (Click here to create)

Sybase ASE option	Possible values	Description
Aliases	-	Enter the alias name and the owner name to which the alias name maps.

### 2.2.7.2.15 Sybase IQ

Displayed options vary depending on the version of Sybase IQ you select in the datastore editor.

Table 2-94: Main window

Sybase IQ option	Possible values	Description
Database version	Currently supported versions	Select the version of Sybase IQ that this datastore accesses. Displayed options in the rest of the datastore editor vary depending on the version selected.



Sybase IQ option	Possible values	Description
Use data source name (DSN)	Check box selected or not selected	<p>Select to use DSN to connect to the database.</p> <p>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 <b>Hostname</b>, <b>Database name</b>, and <b>Port</b>, and <b>Server name</b>.</p> <p>If you select this checkbox, you must fill in <b>Data source name</b></p>
Hostname	Computer name or IP address	<p>Type the computer name or IP address.</p> <p>This option is required if you did not select <b>Use data source name (DSN)</b>.</p>
Database name	Refer to the requirements of your database	<p>Type the name of the database defined in Sybase IQ .</p> <p>This option is required if you did not select <b>Use data source name (DSN)</b>.</p>
Port	Four digit integer Default: 2638	<p>Type the number of the database port.</p> <p>This option is required if you did not select <b>Use data source name (DSN)</b>.</p>
Server name	Refer to the requirements of your database	<p>Type the Sybase IQ database server name.</p> <p>This option is required if you did not select <b>Use data source name (DSN)</b>.</p>
Data source name	Refer to the requirements of your database	<p>Select or type the Data Source Name defined in the ODBC Administrator for connecting to your database.</p> <p>This option is required when you select <b>Use data source name (DSN)</b>.</p>
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 2-95: General

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 <b>Browse</b>	<p>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.</p> <p>If you do not enter a name here, Data Services by default writes the files to the directory <code>&lt;DS_COMMON_DIR&gt;\log\bulkloader</code>.</p>
Overflow file directory	Directory path or click <b>Browse</b>	Enter the location of overflow files written by target tables in this datastore. You can enter a variable for this option.

Table 2-96: Locale

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 2-97: Bulk loader

Sybase IQ option	Possible values	Description
JS and DB on same machine	Yes, No	<p>For some versions of 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).</p> <p>Refer to the following table for how to use this option together with the <b>Use named pipe</b> option and FTP depending on the version of Sybase IQ you are using.</p>
Server working directory	Absolute file path	<p>Type the path to the working directory for the load utility on the computer that runs the Sybase IQ server. If an absolute file path is specified for the <b>FTP host working directory</b> box, then the Server working directory is optional. If the box is left blank, then the software uses the file path to the <b>FTP host working directory</b>.</p> <p>If a virtual file path is specified for the <b>FTP host working directory</b>, then you must enter an absolute file path in the <b>Server working directory</b> box.</p>
Use named pipe	Yes, No	<p>Select Yes to eliminate the need to write a data file to disk, which can improve performance. If a data file is required for Sybase IQ database recovery, select No. Defaults to No.</p> <p>Refer to the following table for how to use this option together with the <b>JS and DB on same machine</b> option and FTP depending on the version of Sybase IQ you are using.</p>

Table 2-98: Supported bulk loader options and methods

JS and DB on same machine	Use named pipe	For Sybase IQ versions earlier than 15.x	For 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 2-99: FTP

Sybase IQ option	Possible values	Description
FTP host name	Computer name, fully qualified domain name, or IP address	For some versions of Sybase IQ, Data Services generates a data file and transfers it via FTP to the database for loading.  Type the name of the 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 Sybase IQ server.  For Windows servers only, you can configure a path to a virtual directory.

Table 2-100: Session

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 2-101: Aliases (Click here to create)

Sybase IQ option	Possible values	Description
Aliases		Enter the alias name and the owner name to which the alias name maps.

### 2.2.7.2.16 Teradata

Table 2-102: 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	<p>Select to use DSN to connect to the database.</p> <p>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 <b>Database server name</b>, <b>Database name</b>, and <b>Port</b>.</p> <p>If you select this checkbox, you must fill in <b>Data source name</b></p>
Database server name	Refer to the requirements of your database	<p>Type the Teradata database server name.</p> <p>This option is required if you did not select <b>Use data source name (DSN)</b>.</p>
Database name	Refer to the requirements of your database	<p>Type the name of the database defined in Teradata.</p> <p>This option is required if you did not select <b>Use data source name (DSN)</b>.</p>
Port	<p>Four digit integer</p> <p>Default: 8888</p>	<p>Type the port number to connect to this database.</p> <p>This option is required if you did not select <b>Use data source name (DSN)</b>.</p>

Teradata option	Possible values	Description
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 <b>Use data source name (DSN)</b> .
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 2-103: General

Teradata option	Possible values	Description
Bulk loader directory	Directory path or click <b>Browse</b>	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 <b>Browse</b>	Specify the directory on the Job Server where Teradata Parallel Transporter-specific files (control and error files) are stored. If this option is left empty, the default location is <code>\$LINK_DIR\log\BulkReader</code> .
Overflow file directory	Directory path or click <b>Browse</b>	Enter the location of overflow files written by target tables in this datastore. You can enter a variable for this option.

Table 2-104: 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."

Teradata option	Possible values	Description
Server code page		See the section "Locales and Multi-Byte Functionality."
Teradata		
Log directory	Directory path or click <b>Browse</b>	The directory in which to write log files.
Tdpld	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 2-105: Session

Teradata option	Possible values	Description
Additional session parameters	A valid SQL statement or multiple SQL statements delimited by semicolons.	<p>Additional session parameters specified as valid SQL statement(s).</p> <p>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:</p> <ul style="list-style-type: none"> <li>On the Teradata server, set the "General" parameter <b>DBSControl</b> to TRUE to allow uncommitted data to be read.</li> <li>In the Data Services Teradata datastore, add the following statement in the "Additional session parameters" field:</li> </ul> <pre>SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;</pre>

Table 2-106: 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.

### 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	<p>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:</p> <ul style="list-style-type: none"> <li>• DB2</li> <li>• Microsoft SQL Server</li> <li>• ODBC</li> <li>• Oracle</li> </ul> <p>For details on configuring the datastore options for JD Edwards applications, refer to the <i>Supplement for J.D. Edwards</i>.</p>
JDE World	<p>For JDE World, Data Services supports the database type ODBC.</p> <p>For details on configuring the datastore options for JD Edwards , applications refer to the <i>Supplement for J.D. Edwards</i>.</p>
Oracle Applications	<p>For details on configuring the datastore options for Oracle applications, refer to the <i>Supplement for Oracle Applications</i>.</p>
PeopleSoft	<p>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:</p> <ul style="list-style-type: none"> <li>• Microsoft SQL Server</li> <li>• Oracle</li> </ul> <p>For more information about PeopleSoft applications, refer to the <i>Supplement for PeopleSoft</i>.</p>



Datastore type	More information
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 <i>Supplement for SAP</i> .
SAP BW Source	
SAP BW Target	
SAP Master Data Services	
Siebel	<p>The Siebel datastore type works with the following database types:</p> <ul style="list-style-type: none"> <li>• DB2</li> <li>• Microsoft SQL Server</li> <li>• Oracle</li> </ul> <p>For more information about Siebel applications, refer to the <i>Supplement for Siebel</i>.</p>
Web Service	The options for the Web Service datastore type are documented in the <i>Integrator's Guide</i> .

After you create a datastore, you can import metadata about the objects, such as tables and functions, into that datastore.

### Related Topics

- [Designer Guide: Datastores, What are datastores?](#)
- [Supplement for JDEdwards: Datastores](#)
- [Supplement for Oracle Applications: Datastores](#)
- [Supplement for PeopleSoft: PeopleSoft Datastores](#)
- [Supplement for PeopleSoft: Reference Information, Datastore](#)
- [Supplement for SAP: Reference Information, Datastore](#)
- [Supplement for SAP: Connecting to SAP Applications, SAP Applications datastores](#)
- [Supplement for SAP: Connecting to SAP NetWeaver Business Warehouse, SAP NetWeaver BW datastores](#)
- [Supplement for Siebel: Datastores](#)
- [Integrator's Guide: Consuming external web services in SAP BusinessObjects Data Services](#)

## 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.

## 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 BusinessObjects 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 Topics**

- [XML schema](#)
- [Rules for importing DTDs](#)

**2.2.9.1 Editor**

Open the DTD editor by double-clicking a DTD name in the object library.

**Related Topics**

- [Designer Guide: Using Document Type Definitions \(DTDs\)](#)

**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, <code>C:\data\test.dtd</code> . 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 BusinessObjects Data Services only imports elements of the format that belong to this node or any sub nodes.

**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 BusinessObjects 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.

## 2.2.9.4 Supported DTD components

SAP BusinessObjects 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.

DTD declaration		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	<p>Supported. The software saves the enumerated values in the Enumeration attribute of the column.</p> <p>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.</p> <p>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.</p>

DTD declaration		Supported
Attribute declaration defaults	#REQUIRED	Supported. The software saves this as the Required attribute with a value of YES and as data type <code>varchar(1024)</code> . When producing 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 <code>varchar(1024)</code> . 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 <code>varchar(1024)</code> . 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 <code>varchar(1024)</code> . 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.

Operator	Description	Supported
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.  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 ()\*.



## 2.2.9.5 Rules for importing DTDs

SAP BusinessObjects 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 (l) 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.

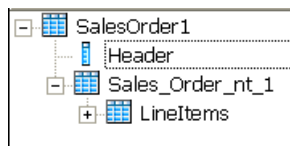
After applying these rules, the software uses two additional rules, except where doing so would allow more than one row for a root element:

- 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.

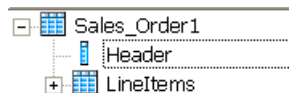
For example, the SalesOrder element might be defined as follows in a DTD:

```
<!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.

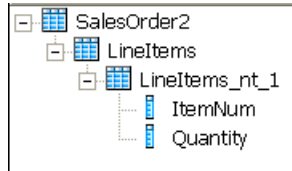


- If a nested table contains one and only one implicit table, then the implicit table can be eliminated and its columns placed directly under the nested table.

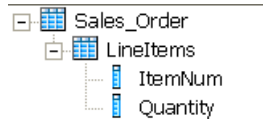
For example, the nested table LineItems might be defined as follows in a DTD:

```
<!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.



### 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 BusinessObjects 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 SAP BusinessObjects 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.

## 2.2.9.7 Error checking

You can control whether SAP BusinessObjects 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.

## 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 BusinessObjects 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.

### 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.

### 2.2.10.1 Notes

- If SAP BusinessObjects 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 /?:!\*[]`.

## 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
<b>Directory</b>	Specify the directory that contains the Excel workbook. You can specify variables or wild cards, but the <b>Import schema</b> button will then be disabled.
<b>File name</b>	Specify the file name for the Excel workbook file. This file contains the schema definition. You can specify variables or wild cards, but the <b>Import schema</b> button will then be disabled.

Format option	Description
<b>Access method</b>	<p>Specifies whether to import a named range (as defined in Excel) or a range from a worksheet:</p> <p><b>Named range</b>—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.</p> <p><b>Note:</b> The drop-down does not display non-contiguous named ranges, even if they exist in the Excel workbook.</p> <p><b>Worksheet</b>—You can select a specific worksheet by either name or ordinal number (select the <b>Number</b> 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.</p> <p><b>Note:</b> 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 (\).</p> <p>The Worksheet method includes the following <b>Range</b> options:</p> <ul style="list-style-type: none"> <li>• <b>All fields</b>—Includes everything from the uppermost left-hand populated cell to the lowest, right-hand cell.</li> <li>• <b>Custom range</b>—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.</li> </ul> <p><b>Note:</b> Subsequently making the software the active window closes this Excel instance.</p> <p><b>Extend range</b>—If you are using a custom range, select this check box to extend the custom range selection to the end of the worksheet.</p>
<b>Code page</b>	<p>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 <i>Reference Guide</i>.</p>
<b>Use first row values as column names</b>	<p>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).</p>

**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
<b>FTP</b>	Select to use FTP to access the data file.
<b>Host</b>	Type the computer (host) name, fully qualified domain name, or IP address of the computer where the data file resides.
<b>User</b>	Type the FTP user name.
<b>Password</b>	Type the FTP user password.
<b>Directory</b>	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 <b>Name</b> field.
<b>File name</b>	Type or browse to the Excel workbook data file <b>Name</b> . You can use variables or wild cards (* or ?). If you leave <b>Directory</b> blank, then type a full path and file name here.
<b>Custom</b>	Select to use a custom executable to access the data file.
<b>Executable</b>	Type the name of the program to read data file.
<b>User</b>	Type the user name.
<b>Password</b>	Type the password.
<b>Arguments</b>	Include any custom program arguments.

### 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 2-117: Performance

Source option	Description
Join rank	<p>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.</p> <p>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.</p> <p>Must be a non-negative integer. Default value is 0.</p> <p>For more information, see the Other Tuning Techniques section in the <i>Performance Optimization Guide</i>.</p>



Source option	Description
Cache	<p>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.</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• <b>Yes:</b> The source is always cached unless it is the outer-most source in a join.</li> <li>• <b>No:</b> The source is never cached.</li> </ul> <p>The default is <b>Yes</b>.</p> <p>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.</p>
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 2-118: 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 <b>Log data conversion warnings</b> is enabled, you can limit how many warnings the software logs. Defaults to <b>{no limit}</b> .
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 2-119: Include file name column

Source option	Description
Include file name column	<p>Determines whether to add a column that contains the source Excel workbook file name in the source output. Defaults to <b>No</b>.</p> <p>Change the value to <b>Yes</b> when you want to identify the source Excel workbook file in situations such as the following:</p> <ul style="list-style-type: none"> <li>You specified a wildcard character to read multiple source Excel workbook files at one time</li> <li>You load from different source workbook files on different runs</li> </ul>
Modify	If the file name is included, this button enables you to modify <b>File name column</b> and <b>Column size</b> .
File name column	<p>If the file name is included, the name of the column that holds the source Excel workbook file name.</p> <p>Defaults to <b>DI_FILENAME</b>.</p>
Column size	<p>If the file name is included, the size (in characters) of the column that holds the source Excel workbook file name.</p> <p>Defaults to <b>100</b>. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.</p>
Include path	<p>If the file name is included, determines whether to include the full path name of the source Excel workbook file.</p> <p>Defaults to <b>No</b>.</p>

Table 2-120: 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 <b>{none}</b> .
Read total	Optionally enter the number of total rows to read starting at the top of the worksheet or after the <b>Skip first</b> value. Defaults to <b>{no limit}</b> .

**Related Topics**

- [Performance Optimization Guide: Other tuning techniques, Join ordering](#)
- [Designer Guide: Embedded Data Flows](#)

## 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 <b>Formats</b> tab in the object library, right-click <b>Flat Files</b> , and select <b>New</b> . The "File Format Editor" appears.
Edit	Use to edit an existing file format template. To open in edit mode, go the <b>Formats</b> tab in the object library, select an existing flat file format, double-click, or right-click and select <b>Edit</b> . 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.
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
<b>General</b>			

Option	Possible values	Description	Mode
Type	Delimited, Fixed width, SAP transport, Unstructured text, Unstructured binary	<p>The format of the data in the text file. Available properties change based on the selected file format type.</p> <p>For information about the SAP transport file format, see the <i>Supplement for SAP</i>.</p>	New, Edit
Name	Any alphanumeric character and under-scores	A descriptive name for the file format. This name appears in the object library.	New
Join rank	Integer greater than or equal to 0	<p>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.</p> <p>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.</p> <p>Must be a non-negative integer. Default value is 0.</p> <p>For more information, see the “Other Tuning Techniques” section in the <i>Performance Optimization Guide</i>.</p>	Source
Cache	Yes, No	<p>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.</p> <p>Options are:</p> <ul style="list-style-type: none"> <li><b>Yes:</b> The source is always cached unless it is the outer-most source in a join.</li> <li><b>No:</b> The source is never cached.</li> </ul> <p>The default is <b>Yes</b>.</p> <p>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.</p>	Source

Option	Possible values	Description	Mode
Adaptable schema	Yes, No	<p>Indicates whether the schema of a delimited file format is adaptable or fixed.</p> <ul style="list-style-type: none"> <li><b>Yes</b> 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.</li> <li><b>No</b> 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.</li> </ul> <p>The default is <b>No</b>. If you select <b>Yes</b>, you must ensure that the selected column and row delimiters do not appear inside the actual data.</p>	New, Edit, Source
Data Alignment	Character, Byte	<p>Indicates how the software will process fixed-width file formats. The default setting is <b>Character</b>. <b>Character</b> indicates that fields in your data are measured as character. All processing will be in character semantics. <b>Byte</b> 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 var-char(30), it means that the column has 30 characters regardless of the number of bytes for each character. In byte semantic, var-char(30) means 30 bytes. Byte semantic recognizes bytes only, it does not recognize characters.</p> <p><b>Note:</b></p> <p>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.</p> <p>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" .</p>	New, Edit

Option	Possible values	Description	Mode
Make port	Yes, No	Indicates whether the file is an embedded data flow port. Choose <b>Yes</b> to make a source or target file an embedded data flow port. The default is <b>No</b> .  For more information, see “Creating embedded data flows” in the <i>Designer Guide</i> .	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 <b>Yes</b> disables the Error Handling section in the Format editor. The default is <b>No</b> .	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 <i>Performance Optimization Guide</i> .  For jobs that process USPS certification tests, the value should be set to <b>{none}</b> .	All
<b>Data File(s)</b>			
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

Option	Possible values	Description	Mode
Root directory	Path name for the file or blank	<p>The directory where the file is located.</p> <p>For added flexibility, you can enter a variable for this option.</p> <p>If you enter a directory name, then enter only the file name for the <b>File</b> property.</p> <p>If you leave the root directory blank, then enter a file name that includes the full path name in the <b>File</b> property.</p>	New, Edit
File name(s)	File name(s), file name(s) including full path name, or blank	<p>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.</p> <p>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:</p> <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> </ul> <p>See “Reading multiple files at one time” in the <i>Designer Guide</i>.</p>	All
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	<p>For unstructured file formats, specifies whether to ignore empty files.</p> <ul style="list-style-type: none"> <li><b>Yes</b> skips empty files.</li> <li><b>No</b> creates a row with NULL data.</li> </ul>	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	<p>Specifies whether the software should delete the file before loading.</p> <ul style="list-style-type: none"> <li><b>Yes</b> indicates that the software should delete the file before loading.</li> <li><b>No</b> indicates that the software should append to the existing file.</li> </ul>	Target

Option	Possible values	Description	Mode
<b>Delimiters</b>			
Column	Tab, Semi-colon, 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
Row within text string	Character, Row delimiter	Defines how the row delimiter is interpreted within a text string. <ul style="list-style-type: none"> <li>• <b>Character:</b> The specified row delimiter is treated as characters within the text string.</li> <li>• <b>Row delimiter:</b> The specified row delimiter is interpreted and defines rows within the text string.</li> </ul>	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 <b>Row within text string</b> setting.	New, Edit
<p><b>Note:</b></p> <p>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.</p> <p>You can use any ASCII characters (including non-printing characters) for column and row delimiters.</p> <p>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.</p>			
<b>Default Format</b>			



Option	Possible values	Description	Mode
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. <ul style="list-style-type: none"> <li><b>Leading:</b> Adds blanks to the left of (before) the data.</li> <li><b>Trailing:</b> Adds blanks to the right of (after) the data.</li> </ul>	New, Edit
Blank trimming	leading, trailing, both, or none	For fixed-width file format sources, trims extra blank spaces before or after the fields. <ul style="list-style-type: none"> <li><b>Leading:</b> Trims blanks from the left of (before) the data.</li> <li><b>Trailing:</b> Trims blanks from the right of (after) the data.</li> </ul>	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

Option	Possible values	Description	Mode
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	<p>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.</p> <p><b>Note:</b> 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.</p> <p>To improve performance you can manually deselect this conversion and validation operation. In the Target File Editor, set <b>Validate decimal data</b> to No.</p>	Target
<b>Input/Output</b>			
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 or No	<p>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 <b>Skipped rows</b> property.</p> <p>When you select <b>Yes</b>, the software does not read data from the first row, and uses data in the first row to determine the file's column names.</p>	New, Edit
Write row header	Yes or No	For file formats using Headers style, indicates whether to write column names in first row of output file.	New, Edit

Option	Possible values	Description	Mode
Write BOM	Yes or 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
<b>Custom Transfer</b>			
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

Option	Possible values	Description	Mode
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
<b>Locale</b>	For more information about locales for sources, targets, and the software's internal processing, see <a href="#">Locales and Multi-byte Functionality</a> .		
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 pairing 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
<b>Error Handling</b>			

Option	Possible values	Description	Mode
Log data conversion warnings	Yes, No	Determines whether to include data-type conversion warnings in the error log. Defaults to <b>Yes</b> .	New, Edit, Source
Log row format warnings	Yes, No	Determines whether to include row-format warnings in the error log. Defaults to <b>Yes</b> .	New, Edit, Source
Log warnings	Yes, No	For unstructured file formats, determines whether to log warnings including: <ul style="list-style-type: none"> <li>• When there are no files in the specified directory.</li> <li>• When no files match the specified filter.</li> <li>• When skipping an irregular file on UNIX (for example, a FIFO, symbolic link, character or block device, or UNIX socket).</li> <li>• When encountering an empty file with <b>Skip empty files</b> set to <b>Yes</b>.</li> </ul>	New, Edit, Source
Maximum warnings to log	Integer greater than 0 or {no limit}	If <b>Log data conversion warnings</b> or <b>Log row format warnings</b> is enabled, you can limit how many warnings the software logs. Defaults to <b>{no limit}</b> .	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 <b>No</b> .	New, Edit, Source
Capture row format errors	Yes, No	Determines whether to capture row-format errors when processing a flat-file source. Defaults to <b>Yes</b> .	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 <b>Yes</b> .	New, Edit, Source
Maximum errors to stop job	Integer greater than 0 or {no limit}	If <b>Capture data conversion errors</b> or <b>Capture row format errors</b> is enabled, you can limit how many invalid rows the software processes before stopping the job. Defaults to <b>{no limit}</b> .	New, Edit, Source
Write error rows to file	Yes, No	Determines whether to write invalid rows to an error file. Defaults to <b>No</b> .	New, Edit, Source

Option	Possible values	Description	Mode
Error file root directory	Directory path or blank	<p>If <b>Write error rows to file</b> is enabled, type the root directory in which to save the error file. For added flexibility, you can enter a variable for this option.</p> <p>If you type a directory path here, then only enter the file name in the <b>Error file name</b> property.</p> <p>If you leave <b>Error file root directory</b> blank, then type a full path and file name in the <b>Error file name</b> property.</p>	New, Edit, Source
Error file name	File name, file name including full path name, or blank	<p>If <b>Write error rows to file</b> is enabled, type the name of the file in which to record the invalid rows.</p> <p>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.</p>	New, Edit, Source
<b>Source Information</b>			
Include file name column	Yes, No	<p>Determines whether to add a column that contains the source file name in the source output. Defaults to <b>No</b>.</p> <p>Change the value to <b>Yes</b> when you want to identify the source file in situations such as the following:</p> <ul style="list-style-type: none"> <li>You specified a wildcard character to read multiple source files at one time</li> <li>You load from different source files on different runs</li> </ul>	Source
Column name		If the file name is included, the name of the column that holds the source file name. Defaults to <b>DI_FILENAME</b> .	Source
Column size		<p>If the file name is included, the size (in characters) of the column that holds the source file name.</p> <p>Defaults to <b>100</b>. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.</p>	Source
Include path		If the file name is included, determines whether to include the full path name of the source file. Defaults to <b>No</b> .	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 <b>Yes</b> for the <b>Write row header</b> 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.
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.

Property	Possible values	Description
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, Prenom, Prenom_Match_Std, Region, Sort-code_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 Topics

- [Supported locales and encodings](#)
- [HDFS file format](#)
- [File multi-threading](#)

## 2.2.12 Function



### 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 BusinessObjects Data Services
- Custom functions you create
- Validation functions that you can import from SAP BusinessObjects 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	<p>Each imported or custom function has a type. For imported and custom functions, right-click the function from the object library and select <b>Properties</b> to view the type. Descriptions and syntax for built-in functions is listed in the function wizard and the smart editor.</p> <p>Some functions also include a Category designation on the "Function" tab of the Properties dialog box.</p>
Enable Parallel Execution	<p>Check box on the Properties dialog box. Enables the software to run stored procedures and custom functions in parallel.</p> <p>This option must be selected in addition to entering a positive number for the parent data flow's degree of parallelism.</p> <p>For more information, see "Degree of parallelism" in the <i>Performance Optimization Guide</i>.</p>
Validation function	Check box on the Properties dialog box indicating if the function is a validation function.

### Related Topics

- [Functions and Procedures](#)

## 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](#).

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
<b>Data File(s)</b>			
NameNode host	Computer name, fully qualified domain name, IP address, or variable	Name of the NameNode computer.  If you use the default settings of <b>default</b> for NameNode host and <b>0</b> 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
<b>Pig</b>			
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.	All
Clean up working directory	Yes, No	Select <b>Yes</b> 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 <b>No</b> 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.	All
<b>Locale</b>			

Option	Possible values	Description	Mode
Code page	<default> us-ascii	For better performance, select <b>&lt;default&gt;</b> (UTF-8) or <b>us-ascii</b> , which are supported by Pig.  For other code pages, Data Services uses HDFS API-based file reading.	All

### Related Topics

- [File format](#)

## 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 Topics

- [Trace logs](#)

- [Monitor logs](#)
- [Error logs](#)

### 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."

Error number prefix	Description
BLKLOAD	Traces bulk loading.
DATAFLOW	Traces the data flow execution.
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 Topics**

- [Performance Optimization Guide: Checking system utilization](#)

## 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	<p>Indicates which object is executing. The path name has the following format:</p> <pre>dfname[_subdataflownumber]/objectname</pre> <p>where</p> <ul style="list-style-type: none"> <li>• <i>dfname</i> is the name of the data flow</li> <li>• <i>_subdataflownumber</i> is the number of the sub data flow if SAP BusinessObjects Data Services split the data flow into multiple sub data flows</li> <li>• <i>objectname</i> is the name of the source, transform, or target that the data flow is processing</li> </ul> <p>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:</p> <pre>/Orders_DF_1/ORDER DETAILS</pre> <p>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.</p> <pre>/Orders_DF_1/TS_ORDER DETAILS_ORDERTEMP</pre> <p>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:</p> <pre>/Orders_DF_2/Query</pre>
State	<p>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.</p>
Row Count	<p>Indicates the number of rows processed through this object. This value is updated based on the <b>Monitor sample rate (# of seconds)</b> set as a debug property.</p>

Entry	Description
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.

### 2.2.14.3 Error logs

The **error log** lists errors generated by SAP BusinessObjects 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.
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.

Error number prefix	Description
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.

## 2.2.15 Message function



### 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.



## 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 BusinessObjects 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.

## 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 BusinessObjects 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.

## 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 Topics

- [Designer Guide: Query](#)
- [Designer Guide: Nested Data](#)

## 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 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.

Attribute	Description
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 Topics

- [Log](#)

## 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.

### 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).

### 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

### 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 BusinessObjects 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.

### 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.

## 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 BusinessObjects 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 Topics

- [Scripting Language](#)

## 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 BusinessObjects 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 Topics

- [Designer Guide: Embedded Data Flows](#)

## 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 BusinessObjects 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	<p>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.</p> <p>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.</p> <p>Must be a non-negative integer. Default value is 0.</p> <p>For more information, see the Other Tuning Techniques section in the <i>Performance Optimization Guide</i>.</p>



Option	Description
Cache	<p>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.</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• <b>Yes:</b> The source is always cached unless it is the outer-most source in a join.</li> <li>• <b>No:</b> The source is never cached.</li> </ul> <p>The default is <b>Yes</b>.</p> <p>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.</p>
Array fetch size	<p>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.</p> <p>This option is available for source tables from DB2, Informix, ODBC, Oracle, and SQL Server datastores.</p> <p>When retrieving a column with an Oracle LONG data type, the software automatically sets <b>Array Fetch Size</b> to 1. If a column has an Oracle LONG data type, the software can only retrieve one row at a time.</p>

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 Topics

- [Designer Guide: Creating embedded data flows](#)
- [Performance Guide: Using parallel execution](#)
- [Performance Optimization Guide: Other tuning techniques, Join ordering](#)

### 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, and Microsoft SQL Server changed-data capture options. Options vary by source type.

Option	Description
CDC subscription name	<p>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.</p> <p>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.</p> <p>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.</p> <p>This value is required for the Microsoft SQL server Change Tracking and Replication Server methods.</p>
Enable check-point	<p>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.</p> <p>By default, check-points are not enabled.</p>
Automatically delete rows after reading	<p>For Microsoft SQL Server only.</p> <p>If the option is cleared, no rows are deleted from the CDC table.</p> <p>If the option is selected, the behavior is different depending on the CDC method used:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> </ul>
Get before-image for each update row	<p>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.</p> <p>By default, only after-images are retrieved.</p>

**Related Topics**

- [Designer Guide: Capturing Changed Data](#)

### 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 BusinessObjects 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 Topics**

- [Table source](#)
- [File format](#)

### 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 <b>True</b> 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 example, if you have four CPUs, enter 4 for this option.

**Related Topics**

- [Performance Optimization Guide: Using Parallel Execution, File multi-threading](#)

### 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 BusinessObjects 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 <b>Table name</b> box displays the name that you entered when you created the persistent cache table. You cannot change this name.
Table owner	The <b>Table owner</b> box displays the owner that you entered when you created the persistent cache table. You cannot change this name.
Datastore name	The <b>Datastore name</b> box displays the name that you entered when you created the persistent cache. You cannot change this name.
Database type	The <b>Database type</b> box displays the <b>Persistent Cache</b> option, which you cannot change.

#### Related Topics

- [Table source](#)
- [Designer Guide: Creating embedded data flows](#)

### 2.2.21.6 SAP sources

SAP sources include Open Hub tables and SAP extractors.

For details on SAP sources, refer to the *SAP BusinessObjects Data Services Supplement for SAP*.

#### Related Topics

- [Supplement for SAP: Open Hub Table source](#)
- [Supplement for SAP: Extractor](#)

### 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 availability 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
<b>General</b>	
Clean up bulk reader directory after export	Select <b>Yes</b> to delete all files in the bulk reader directory after successfully exporting.  Select <b>No</b> 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 <b>Number of export operator instances</b> and <b>Parallel process threads</b> 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 <b>Maximum number of sessions</b> and <b>Parallel process threads</b> 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.
<b>Data handling</b>	
Block size	Specifies the block size (in bytes) when returning data to the client.
Data encryption	Select <b>Yes</b> to enable full security encryption of SQL requests, responses, and data. Select <b>No</b> to disable encryption.
Query band session	Specifies a user-defined query band expression to be set for every SQL session.
<b>Notification</b>	
Level	Indicates the level at which certain events are reported: <ul style="list-style-type: none"> <li>Off: No notifications. Default.</li> <li>Low: Notifications occur for Initialize, CLIV2/DBS Error, Exit.</li> <li>Medium: Notifications occur for all events except File or OUT-MODE Open and Statement Fetch Begin and End.</li> <li>High: Notifications occur for all events.</li> </ul>
Method	Specifies the method for reporting events: <ul style="list-style-type: none"> <li>None: No event logging. Default.</li> <li>Message: Send events to a log (for example the EventLog on Windows).</li> <li>Exit: Send the events to a user-defined notify exit routine and to the system log.</li> </ul>
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.
<b>Trace</b>	

Option	Description
Level	<p>Specifies the type(s) of diagnostic messages each instance of the driver writes to a log file. API mode writes to external log files and Export Operator mode writes to public or private logs.</p> <ul style="list-style-type: none"> <li>• CLI: Activates the tracing function for CLIV2-related activities.</li> <li>• PX: Activates the tracing function for activities involving the common library.</li> <li>• Oper: Activates the tracing function for driver-specific activities.</li> <li>• Notify: Activates the tracing function for activities related to the Notify feature.</li> </ul>
Tracing file	For API mode, specifies the name of the external log file used for trace messages.
<b>Miscellaneous</b>	
Parallel process threads	<p>Specifies the number of threads for parallel processing, which can improve performance by maximizing CPU usage on the Job Server computer.</p> <p>Use this parameter in conjunction with <b>Maximum number of sessions</b> and <b>Number of export operator instances</b> 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.</p>
Logon mechanism	<p>Specifies which logon mechanism to use:</p> <ul style="list-style-type: none"> <li>• Kerberos 5</li> <li>• NT Lan Manager</li> <li>• Lightweight Directory Access Protocol</li> <li>• Simple and Protected GSSAPI Negotiation Mechanism</li> </ul>
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 Topics

- [Performance Optimization Guide: Bulk Loading and Reading, Bulk loading and reading in Teradata](#)
- [Performance Optimization Guide: Using Parallel Execution, File multi-threading](#)
- [Teradata](#)

- [Teradata target table options](#)

## 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	<p>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.</p> <p><b>Note:</b> 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.</p>
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	<p>Determines whether to include a column in the source output that contains the source XML file name. Defaults to <b>No</b>.</p> <p>Change the value to <b>Yes</b> when you want to identify the source XML file in situations such as the following:</p> <ul style="list-style-type: none"> <li>• You specified a wildcard character to read multiple source XML files at one time</li> <li>• You load from different source XML files on different runs</li> </ul>
Modify	If the file name is included, this button enables you to modify <b>File name column</b> and <b>Column size</b> .
File name column	If the file name is included, the name of the column that holds the source file name. Defaults to <b>DI_FILENAME</b> .
Column size	<p>If the file name is included, the size (in characters) of the column that holds the source file name.</p> <p>Defaults to <b>100</b>. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.</p>



Option	Description
Include path	If the file name is included, determines whether to include the full path name of the source file. Defaults to <b>No</b> .

### Related Topics

- [Table source](#)

## 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	<p>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.</p> <p><b>Note:</b> 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. A variable can also be used.</p>
Enable validation	<p>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.</p> <p>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 BusinessObjects 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.</p>
<i>implied</i> : 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 Topics

- [Table source](#)

## 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 BusinessObjects 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 SAP BusinessObjects Data Services and SAP BusinessObjects Universe Builder (UB) 1.1.  A configurable field for the logical Business_Name used in SAP BusinessObjects Universe Builder. SAP BusinessObjects Data Services can extract, transform and load data while transferring this information intact.

Table Attribute	Description
Business_Description	<p>Supports metadata exchanged between SAP BusinessObjects Data Services and SAP BusinessObjects Universe Builder (UB) 1.1.</p> <p>A configurable, business-level description of the table. SAP BusinessObjects Data Services transfers this information separately and adds it to any SAP BusinessObjects Universe Builder Class description.</p>
Table_Usage	A configurable label field. Use it to mark a table as <code>fact</code> or <code>dimension</code> 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.
Date_Created	The date that the object was created.
Estimated_Row_Count	<p>A configurable estimate of the table size used in calculating the order in which tables are read to perform join operations.</p> <p>Used for SAP tables only.</p>
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 <code>TABLE</code> 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.

Table Attribute	Description
Loader_Is_Template_Table	If <b>YES</b> , 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 <b>YES</b> , 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 <b>YES</b> , indicates that you modified the partitions in this table using the software after you imported the table's metadata.

#### Related Topics

- [Performance Optimization Guide: Overview of parallel execution](#)
- [Template table](#)

### 2.2.22.1 Column attributes for tables

SAP BusinessObjects Data Services also supports column attributes for tables.

#### 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	<p>Supports metadata exchanged between SAP BusinessObjects Data Services and SAP BusinessObjects Universe Builder (UB) 1.1.</p> <p>Enter <i>Dimension</i>, <i>Measure</i>, or <i>Detail</i> as a value for a corresponding object in SAP BusinessObjects Universe Builder.</p> <ul style="list-style-type: none"> <li>• If set to <i>Dimension</i>, the corresponding object created in SAP BusinessObjects Universe Builder is of qualification type <b>Dimension</b>.</li> <li>• If set to <i>Measure</i>, the corresponding object is of qualification type <b>Measure</b>.</li> <li>• If set to <i>Detail</i>, the corresponding object is of qualification type <b>Detail</b> and requires you to set a value for <b>Associated_Dimension</b>.</li> </ul>
Associated_ Dimension	Set this value only if <b>Column_Usage</b> is set to <i>Detail</i> . The value must be in the format: <code>table.column</code> . The Detail column is created under the Dimension column you specify.
Acta_autojoin	Generated by SAP BusinessObjects 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 Topics

- [Designer Guide: Attributes that support metadata exchange](#)
- [Attributes for DTDs](#)
- [Attributes supported for XML schemas](#)

## 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 BusinessObjects 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 Topics**

- [Designer Guide: Embedded Data Flows](#)
- [Designer Guide: Creating embedded data flows](#)

### **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 BusinessObjects 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 Topics**

- [File format](#)

### **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.

#### **Note:**

You cannot update a persistent cache table. If the data within it changes, you must recreate it and load it.

Table 2-145: Target

Option	Description
Make port	Select this check box to make the target table an embedded data flow port.
Table name	The <b>Table name</b> box displays the name that you entered when you created the persistent cache table. You cannot change this name.
Table owner	The <b>Table owner</b> box displays the owner that you entered when you created the persistent cache table. You cannot change this name.
Datastore name	The <b>Datastore name</b> box displays the name that you entered when you created the persistent cache. You cannot change this name.
Database type	The <b>Database type</b> box displays <code>Persistent_Cache</code> which you cannot change.

Table 2-146: Options

Option	Description
Column comparison	Specifies how the input columns are mapped to persistent cache table columns. There are two options: <ul style="list-style-type: none"> <li><b>compare_by_position</b>: SAP BusinessObjects Data Services disregards the column names and maps source columns to target columns by position</li> <li><b>compare_by_name</b>: the software maps source columns to target columns by name</li> </ul>
Include duplicate keys	Specifies whether or not to include duplicate keys in the persistent cache. Defaults to selected.

Table 2-147: Keys

Option	Description
Key column	<p>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.</p> <p>To change the order of the columns in the key, use one of the following options:</p> <ul style="list-style-type: none"> <li>Right-click the column and select <b>Move Up</b> or <b>Move Down</b>.</li> <li>Select the column and click the down or up arrow in the top right corner of the <b>Keys</b> tab.</li> </ul> <p>To remove a column, use one of the following options:</p> <ul style="list-style-type: none"> <li>Right-click the column and select <b>Delete</b>.</li> <li>Select the column and click the delete icon in the top right corner of the <b>Keys</b> tab.</li> </ul>



## Related Topics

- [Designer Guide: Embedded Data Flows](#)

### 2.2.23.3 Target tables

You can add a table to a data flow diagram as a target if SAP BusinessObjects 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.

#### 2.2.23.3.1 Common target table options

This table describes the options common to all supported database types.

Table 2-148: Target tab

Option	Description
<b>Make port</b>	Select this check box to make the target table an embedded data flow port.

Option	Description
Database type	<p>Select an item in the <b>Database type</b> 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.</p> <p>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.</p> <p>The software allows you to use target table editor option values from any datastore configuration:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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 <b>Use values from</b> box in the Create New Configuration dialog (a sub-dialog of the datastore editor).</li> <li>• If you also select the <b>Restore values if they already exist</b> check box (in the Create New Configuration dialog), SAP BusinessObjects 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 configuration you select from the <b>Use values from</b> option. For example, suppose you set a configuration for Oracle 8i. When you edit the target table editor options, you change the <b>Rows Per Commit</b> 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 <b>Use values from</b> option to Oracle 8i. In this case, the target table editor settings for SQL Server inherit the value 500 for <b>Rows per Commit</b> because this was the value set in the Oracle 8i configuration.</li> </ul> <p>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.</p>

Table 2-149: Options tab

Option	Description
<b>Rows per commit</b>	<p>Specifies the transaction size in number of rows.</p> <p>For example, if set to 1000, the software sends a commit to the underlying database every 1000 rows.</p> <p>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.</p> <p>For example, suppose you set <b>Rows per commit</b> to 3 and specified an insert trigger, where an incoming insert statement is converted into 5 statements. <b>Rows per commit</b> would automatically be extended to 5 to accommodate each insert trigger statement in a single transaction.</p> <p>This option is not available for targets in real time jobs.</p>
<b>Delete data from table before loading</b>	<p>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, Sybase) or sends a DELETE statement to databases that do not support TRUNCATE. Default setting is not selected.</p> <p>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.</p>
<b>Drop and re-create table</b>	<p>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.</p>
General	
<b>Column comparison</b>	<p>Specifies how the input columns are mapped to output columns. There are two options:</p> <p><b>Compare by position:</b> The software disregards the column names and maps source columns to target columns by position.</p> <p><b>Compare by name:</b> The software maps source columns to target columns by name.</p> <p>Validation errors occur if the data types of the columns do not match.</p>

Option	Description
<b>Number of loaders</b>	<p>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.</p> <p>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.</p> <p>For example, if you choose a <b>Rows per commit</b> of 1000 and set the <b>Number of Loaders</b> 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.</p>
<b>Enable Partitioning</b>	<p>(Displayed only if the target table data is partitioned)</p> <p>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:</p> <ul style="list-style-type: none"> <li>• Number of Loaders</li> <li>• Enable Partitioning</li> <li>• Transactional Loading</li> </ul> <p><b>Note:</b> If you set <b>Enable Partitioning</b> to Yes and <b>Include in transaction</b> to Yes, the <b>Include in transaction</b> setting overrides the <b>Enable Partitioning</b> option.</p>
Error handling	
<b>Use overflow file</b>	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.
<b>File name</b> <b>File format</b>	<p>These options are available only when you select Yes for the <b>Use overflow</b> 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).</p> <p>You can enter a variable for the file name.</p>
Update control	
<b>Use input keys</b>	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.
<b>Update key columns</b>	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.

Option	Description
Auto correct load	<p>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.</p> <p><b>Note:</b> This option is not available for targets in real time jobs or target tables that contain LONG column(s).</p> <p>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 <b>Use input keys</b> 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.</p> <p>If a matching row does not exist, a new row is inserted, regardless of other options.</p> <p>If a matching row exists, the row is updated depending on the values of <b>Ignore columns with value</b>, and <b>Ignore columns with null</b>:</p> <ul style="list-style-type: none"> <li>• When the column data from the source matches the value in <b>Ignore columns with value</b>, 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.</li> <li>• When the <b>Ignore columns with null</b> 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.</li> </ul> <p>For supported databases, when the <b>Allow merge or upsert</b> 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 <b>Ignore columns with value</b> value if completed in the target editor, and the <b>Ignore columns with null</b> Yes/No setting.</p>

Option	Description
<p><b>Allow merge or upsert</b> (This is available for Microsoft SQL Server 2008 and higher only)</p>	<p>Specifies whether the Optimizer may use a MERGE statement to improve the performance of auto correct load functionality.</p> <p><b>Yes:</b> Allows the Optimizer to consider using a MERGE statement during an auto correct load operation.</p> <p>If the Optimizer does not use a MERGE statement, it uses a T-SQL block to identify, insert, and update rows.</p> <p><b>No:</b> The Optimizer will not use a MERGE statement to improve auto correct load performance.</p> <p>The default is Yes.</p> <p><b>Note:</b> 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.</p>
<p><b>Ignore columns with value</b></p>	<p>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.</p>
<p><b>Ignore columns with null</b></p>	<p>Select Yes if you do not want NULL source columns updated in the target table during auto correct loading.</p> <p>This option is only available when you select Yes for the <b>Auto correct load</b> option.</p>
Transaction control	

Option	Description
<b>Include in transaction</b>	<p>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.</p> <p>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.</p> <p>The tables must be from the same datastore.</p> <p>If you choose to enable transactional loading, other options are not available:</p> <ul style="list-style-type: none"><li>• Rows per commit</li><li>• Use overflow file, and overflow file specification</li><li>• Number of loaders</li><li>• Enable partitioning</li><li>• Bulk loader options</li><li>• Pre load commands</li><li>• Post load commands</li></ul> <p>The software does not push down a complete operation to the database if transactional loading is enabled.</p>
<b>Transaction order</b>	<p>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.</p> <p>In Data flow view, the specified transaction order number appears nearer to the corresponding target.</p>

Option	Description
<b>Commit at end of INSERT...SELECT</b>	<p>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 <b>Rows per commit</b> to create a smaller transaction size than the single commit per job default behavior.</p> <p>If your transaction log size is too small for a single transaction of this type, set a commit size when the following applies:</p> <ul style="list-style-type: none"> <li>the job has a source and target that use the same datastore</li> <li>the job has an Oracle target table</li> <li>the software is optimizing and executing the job by pushing down the read operation to the Oracle target table host.</li> </ul> <p>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.</p> <ul style="list-style-type: none"> <li>the job failed with an Oracle transaction log full error.</li> </ul> <p>Troubleshooting:</p> <ol style="list-style-type: none"> <li>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 <b>Validate &gt; Display Optimized SQL.</b>)</li> <li>2. If so, set the Commit at end of INSERT...SELECT to No and enter a value for <b>Rows per commit.</b></li> </ol> <p>If you use this option, expect to see a decrease in performance.</p>



Option	Description
<b>Use NVARCHAR for VARCHAR columns in supported databases</b>	<p>The software creates <i>nvarchar</i> columns in the template table for all <i>varchar</i> columns in the input schema of the data flow. The data type displays as <i>varchar</i> in the Designer, and, when supported by the DBMS, as <i>nvarchar</i> in the database table.</p> <p>The following database management systems do not support the <i>nvarchar</i> data type:</p> <ul style="list-style-type: none"> <li>• DB2 (non-UTF-8)</li> <li>• Oracle 8.x</li> <li>• Informix</li> <li>• Sybase ASE</li> <li>• Sybase IQ</li> </ul> <p>For these DBMSs, the software creates columns with <i>varchar</i> data types and increases the column size using a codepage conversion factor based on the client code page defined in the datastore.</p> <p><b>Caution:</b></p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• Data truncation occurs when the column size of the source exceeds the maximum size allowed by the target DBMS.</li> </ul>

### 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.

**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 Topics**

- [Designer Guide: Embedded data flows](#)
- [Message processing](#)

**2.2.23.3.2 DB2 target table options**

The following table contains option and description information specific to DB2 target tables.

Table 2-150: Target tab

Option	Description
<b>Make port</b>	Specifies the table as an embedded data flow port.

Table 2-151: Options tab

Option	Description
<b>Allow Merge or upsert</b>	<p>Specifies whether the Optimizer may use a MERGE statement to improve the performance of auto correct load functionality.</p> <p><b>Yes:</b> Allows the Optimizer to consider using a MERGE statement during an auto correct load operation.</p> <p>If the Optimizer does not use a MERGE statement, it uses a stored procedure to identify, insert, and update rows.</p> <p><b>No:</b> The Optimizer will not use a MERGE statement to improve auto correct load performance.</p> <p>The default is Yes.</p>

Table 2-152: Bulk loader tab

Option	Description
<b>Bulk load</b>	<p>Indicate the bulk load method. Choose one of the following options:</p> <p><b>CLI load:</b> 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.</p> <p><b>Import:</b> 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.</p> <p><b>Load:</b> Use the DB2 bulk load utility. The load utility improves performance over the import utility by writing data directly into the data file.</p> <p><b>No:</b> Do not bulk load data.</p>

Option	Description
<b>Generate files only</b>	<p>This option is available only when <b>Bulk load</b> is set to <b>import</b> or <b>load</b>.</p> <p>Select this check box to generate a data and control file. Rather than loading data into the target shown in the data flow, SAP BusinessObjects Data Services generates a control file 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.</p> <p>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 <code>&lt;DS_COMMON_DIR&gt;\log\bulkloader</code> directory.</p> <p>To load the data, you must manually copy the files to the remote system and start the bulk load execution.</p> <p>When you select this check box, only the <b>Text delimiter</b> and <b>Column delimiter</b> options are available.</p>
<b>Clean up bulk loader directory after load</b>	<p>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 <b>Connections</b> tab in the datastore definition, the software writes the files in the <code>&lt;DS_COMMON_DIR&gt;\log\bulkloader</code> directory.</p> <p>When this option is selected, the software deletes the following files after each bulk load unless an error has occurred:</p> <ul style="list-style-type: none"> <li>• Message file (<code>.log</code> file name) that DB2 creates for the import, load, or CLI load</li> <li>• Control file (<code>.ctl</code>) that the software generates only when <b>Bulk load</b> is set to <b>import</b> or <b>load</b></li> <li>• Data file (<code>.dat</code>) that the software generates only when <b>Bulk load</b> is set to <b>import</b> or <b>load</b></li> <li>• Bad file (<code>.bad</code>) that DB2 generates only when <b>Bulk load</b> is set to <b>load</b> and the <b>Data file on client machine</b> option is not checked.</li> </ul> <p>If the <b>Data file on client machine</b> option is checked, DB2 creates the <code>.bad</code> file on the DB2 server working directory (specified in the <b>DB2 Properties</b> tab of the datastore definition). In this case, the software does not delete the <code>.bad</code> file when the bulk load completes.</p>

Option	Description
<b>Mode</b>	<p>Specify the mode for loading data in the target table. Available modes depend on the bulk load method.</p> <p>Available modes when <b>Bulk load</b> is set to <b>import</b>:</p> <p><b>Insert</b>: 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.</p> <p><b>Insert-update</b>: 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.</p> <p><b>Replace</b>: Deletes all existing records in the table, then adds new records.</p> <p><b>Truncate</b>: Deletes all existing records in the table, then adds new records.</p> <p>Available modes when <b>Bulk load</b> is set to <b>CLI load</b> or <b>load</b>:</p> <p><b>Insert</b>: Appends the new records into the target table.</p> <p><b>Replace</b>: Deletes the existing records, and then inserts the loaded data.</p>
<b>Rows per commit</b>	<p>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.</p> <p>This option is available only when <b>Bulk load</b> is set to <b>import</b>.</p>
<b>Save count</b>	<p>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.</p> <p>This option is available only when <b>Bulk load</b> is set to <b>load</b>.</p>
<b>Warning row count</b>	<p>Enter the number of warnings allowed for each load operation.</p> <p>This option is available only when <b>Bulk load</b> is set to <b>CLI load</b> or <b>load</b>.</p>
<b>Text delimiter</b>	<p>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 "\".</p> <p>This option is available only when <b>Bulk load</b> is set to <b>import</b> or <b>load</b>.</p>
<b>Column delimiter</b>	<p>Enter a single-character column delimiter. The default value is a comma (,).</p> <p>This option is available only when <b>Bulk load</b> is set to <b>import</b> or <b>load</b>.</p>

Option	Description
<b>Maximum bind array</b>	<p>Enter the maximum number of rows extracted or transformed before the software sends the data to the DB2 table or view.</p> <p>If you do not enter a value, the software uses the default DB2 CLI Loader value which is 10000.</p> <p>This option is available only when <b>Bulk load</b> is set to <b>CLI load</b>.</p>
<b>Exception table name</b>	<p>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.</p> <p>This option is available only when <b>Bulk load</b> is set to <b>CLI load</b> or <b>load</b>.</p>
<b>Recoverable</b>	<p>Select this check box to support data recovery through the DB2 roll-forward recovery feature.</p> <p>When this option is not selected, you cannot recover from failure using DB2 roll-forward.</p> <p>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 (<b>Copy target directory</b>). Select this option only if your DB2 target database is roll-forward enabled.</p> <p>This option is available only when <b>Bulk load</b> is set to <b>CLI load</b> or <b>load</b>.</p>
<b>Copy target directory</b>	<p>Enter the directory where copy files are stored when the <b>Recoverable</b> option is enabled. Only local media is supported.</p> <p>This option is available only when <b>Bulk load</b> is set to <b>CLI load</b> or <b>load</b>.</p>
<b>Data file on client machine</b>	<p>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:</p> <ul style="list-style-type: none"> <li>• You must use DB2 version 7.x or later.</li> <li>• The target DB2 cannot be a DB2 enterprise (extended edition environment).</li> <li>• The target table and database must not be partitioned.</li> </ul> <p>This option is applicable only if the software and DB2 are on different servers.</p> <p>This option is available only when <b>Bulk load</b> is set to <b>load</b>.</p>

### Related Topics

- [Common target table options](#)

### 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
<b>Append</b>	True, False	Select <b>True</b> to append new data to the table or partition. Select <b>False</b> to delete all existing data, then add new data.
<b>Clean up working directory</b>	True, False	Select <b>True</b> to delete the working directory after the job completes successfully.
<b>Dynamic partition</b>	True, False	Select <b>True</b> for dynamic partitions. Hive evaluates the partitions when scanning the input data. Select <b>False</b> for static partitions. Only all-dynamic or all-static partitions are supported.
<b>Number of loaders</b>	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 number of loaders is greater than one is known as “parallel loading”. You can specify any number of loaders. The default is 1.

### 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 2-153: Bulk Loader Options

Option	Description
Bulk load	Select to use HP Neoview bulk loading options to write the data.



Table 2-154: Options

Option	Description
Clean up bulk loader directory after load	<p>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 BusinessObjects Data Services does not delete script and data files. Errors usually occur when:</p> <ul style="list-style-type: none"> <li>• There is a syntax error in the script.</li> <li>• Error tables are not empty. Error tables contain rows that cannot be inserted into the target table due to data conversion or constraint violation.</li> </ul>
Column delimiter	<p>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 &amp;), except for the following: any alphabetic characters a-z, A-Z, decimal numbers 0-9, carriage return (/10) or line feed (/13).</p> <p><b>Note:</b> The software uses decimal numbers, and HP Neoview uses octal numbers for delimiter values.</p>
File option	<p>Specify the staging type before loading data into the target:</p> <ul style="list-style-type: none"> <li>• Data file</li> <li>• Named pipe</li> </ul>
Generates files only	<p>When this option selected, the software generates a data file and a script file, and ignores the <b>Number of loaders</b> 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.</p> <p>The software writes the data and control files in the bulk loader directory (default value is &lt;DS_COMMON_DIR&gt;\log\bulkloader) specified in the datastore definition. The naming conventions for these files is: &lt;DatastoreName_OwnerName_TableName_PID_n&gt;.ctl and &lt;DatastoreName_OwnerName_TableName_PID_n&gt;.dat where:</p> <ul style="list-style-type: none"> <li>• OwnerName is the table owner</li> <li>• TableName is the target table</li> <li>• PID is the process ID</li> <li>• n is a positive integer, optionally used to guarantee that pre-existing files are not overwritten</li> </ul>
Mode	<p>Specify the mode for loading data in the target table:</p> <ul style="list-style-type: none"> <li>• <b>Append:</b> Adds new records to the table</li> <li>• <b>Truncate:</b> Deletes all existing records in the table, then inserts the loaded data as new records</li> </ul>

Option	Description
SQL Operation	Specify the type of SQL operation to perform: <ul style="list-style-type: none"> <li>• <b>Insert:</b> Places data into a Neoview table.</li> <li>• <b>Update:</b> Updates data in a Neoview table.</li> <li>• <b>Upsert:</b> Updates the data in a Neoview table if the data exists, otherwise it inserts the data.</li> </ul>
Text delimiter	Specify a character used to delimit (enclose) char or varchar data. Valid values are single quotes, double quotes, or blank (default).

Table 2-155: 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 <code>datastore_schema_tablename_PID_n_badrecord.dat</code> in the directory listed in <code>&lt;DS_COMMON_DIR&gt;\log\bulkloader</code> .
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: <ul style="list-style-type: none"> <li>• update trigger on a table with primary key, enable triggers is before update, and operation must be set to update.</li> <li>• update trigger on a table with multiple column primary key, enable triggers is before update, and operation must be set to update.</li> </ul>
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 <code>datastore_schema_tablename_PID_n_failedrecord.dat</code> in the directory listed in <code>&lt;DS_COMMON_DIR&gt;\log\bulkloader</code> .
Force staging	Specify whether to write rows to an internal staging table before populating the target tables.  <b>Note:</b> Using staging tables increases the performance for load operations. However, if the load fails, it may be more difficult to recover.

Option	Description
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 <b>sorted by primary key</b> option and the <b>truncate</b> option to true, and the <b>force staging</b> 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. <b>Note:</b> 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.
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 Topics

- [Common target table options](#)

#### 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](#).

**Note:**

Commit at the end of INSERT..SELECT option is not applicable.

Table 2-156: Options

Option	Description
Drop and re-create table	Drops the existing table and creates a new one with the same name before loading.

Table 2-157: Bulk Loader Options

Option	Description
Bulk load	Select this check box to use Informix bulk loading options to write the data.
Generate files only	<p>Select this check box to generate a data and control file. Rather than loading data into the target shown in the data flow, SAP BusinessObjects 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.</p> <p>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 <code>&lt;DS_COMMON_DIR&gt;\log\bulkloader</code> directory.</p> <p>You need to copy the files to the remote system manually. The files names are <code>tablename.ctl</code> and <code>tablename.dat</code>, where <code>tablename</code> is the name of the target table.</p>
Lock table	Select this check box to lock the table for the duration of the load.
Clean up bulk loader directory after load	<p>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:</p> <ul style="list-style-type: none"> <li>• Control file</li> <li>• Log file</li> <li>• Bad file</li> </ul>

Option	Description
Mode	Select the mode for loading data in the target table: <ul style="list-style-type: none"><li>• <b>Append:</b> Adds new records to the table.</li><li>• <b>Replace:</b> Deletes all existing records in the table and then adds new records.</li></ul>
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 Topics**

- [Common target table options](#)

**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](#).

Table 2-158: Options

Option	Description
<b>Allow merge or upsert</b> (This is available for Microsoft SQL Server 2008 and higher only)	<p>Specifies whether the Optimizer may use a MERGE statement to improve the performance of auto correct load functionality.</p> <p><b>Yes:</b> Allows the Optimizer to consider using a MERGE statement during an auto correct load operation.</p> <p>If the Optimizer does not use a MERGE statement, it uses a T-SQL block to identify, insert, and update rows.</p> <p><b>No:</b> The Optimizer will not use a MERGE statement to improve auto correct load performance.</p> <p>The default is Yes.</p> <p><b>Note:</b> When the data flow contains a Query transform with an <b>Order by</b> clause, the Optimizer always uses a T-SQL block to identify, insert, and update rows.</p>

Table 2-159: Bulk loader options

Option	Description
<b>Bulk load</b>	Select this check box to use Microsoft SQL Server bulk loading options to write the data.
<b>Rows per commit</b>	Select this option to load a specified number of rows per commit.
<b>Mode</b>	Specify the mode for loading data in the target table: <ul style="list-style-type: none"> <li>• <b>Append:</b> Adds new records to the table.</li> <li>• <b>Truncate:</b> Deletes all existing records in the table, and then adds new records.</li> </ul>
<b>Maximum rejects</b>	<p>When the value for <b>Maximum rejects</b> 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.</p> <p>A rejected row contains data that does not match the expected format or information specified for the table being loaded.</p>
<b>Network packet size</b>	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.

### 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).

**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 2-160: Bulk Loader options tab

Option	Description
Bulk load	<p>Select a bulk-loading method:</p> <p><b>File:</b> 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.</p> <p><b>Named pipe:</b> 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.</p> <p><b>None:</b> Does not use bulk loading (default).</p>
Mode	<p>Specify the mode for loading data to the target table:</p> <p><b>Append:</b> Adds new records to table (default).</p> <p><b>Truncate:</b> Deletes all existing records in table then adds new records.</p>
Update method	<p>Specify how to apply UPDATE SQL.</p> <p><b>Delete-insert:</b> 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.</p> <p><b>Note:</b> Do not use this option if the update rows do not contain data for all columns in the target table because SAP BusinessObjects Data Services will replace missing data with NULLs.</p> <p><b>Update:</b> Issues an UPDATE to the target table.</p>
Maximum rejects	<p>Enter the maximum number of error records allowed before the software terminates the job. Default is 10.</p>
Commit size	<p>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 <b>Rows per commit</b> setting.</p>

Option	Description
Text delimiter	<p>Enter the character used to delimit (enclose) char or varchar data. Valid values are single quotes, double quotes, or blank (default).</p> <p>If you expect the data to contain single or double quotes, use an Escape character.</p>
Field delimiter	<p>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.</p>
Generate files only	<p>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.</p>
Escape character	<p>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.)</p> <p>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.</p> <p><b>Note:</b> The null indicator string is not case sensitive so if you have varchar data as NULL (case insensitive) the escape character should be set.</p> <p>Default is cleared because this option can degrade job execution performance significantly.</p>
Clean up bulk loader directory after load	<p>Select this check box to delete all bulk-loader-related files (for example dat, sql, nzlog, nzbad, etc.) after the load completes successfully.</p> <p>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.</p> <p>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.</p>

### Related Topics

- [Common target table options](#)



### 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](#).

Table 2-161: 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 <b>Bulk load</b> is set to <b>import</b> .

#### Related Topics

- [Netezza target table options](#)

### 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](#).

Table 2-162: Options

Option	Description
<b>Allow merge</b>	Specifies whether the Optimizer may use a MERGE statement to improve the performance of auto correct load functionality.  <b>Yes:</b> 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.  <b>No:</b> The Optimizer will not use a MERGE statement to improve auto correct load performance.  The default is Yes.

Table 2-163: Bulk loader options tab

Option	Description
<b>Bulk load</b>	<p>Select a bulk loading method. Use database tools to load data in bulk instead of using SQL statements.</p> <p><b>API:</b> 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.</p> <p><b>File:</b> 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</p> <p><b>None:</b> Allows you to use normal load functionality. See the Options tab.</p>
<b>Mode</b>	<p>Specify the mode for loading data in the target table.</p> <p>Available modes when Bulk load is set to File:</p> <p><b>Append:</b> Adds new records containing the loaded data.</p> <p><b>Insert:</b> 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.</p> <p><b>Replace:</b> Deletes all existing records in the table, and then adds new records containing the loaded data.</p> <p><b>Truncate:</b> Deletes all existing records in the table, and then adds new records containing the loaded data. This mode does not execute any delete triggers.</p> <p>Available modes when Bulk load is set to API:</p> <p><b>Append:</b> Adds new records containing the loaded data.</p> <p><b>Truncate:</b> Deletes all existing records in the table, and then adds new records containing the loaded data.</p>
<b>Rows per commit</b>	<p>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.</p> <p>Bulk loading is not available for targets in a real-time jobs.</p>
<b>Maximum rejects</b>	<p>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.</p>

Option	Description
<b>Recoverable</b>	<p>Select this check box to log direct-path information about the loaded data in the Oracle redo log.</p> <p><b>Note:</b> All conventional loads are automatically specified as recoverable.</p>
<b>Use local index to rebuild primary key</b>	(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.
<b>SQL *Loader version</b>	(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.
<b>Text delimiter</b>	(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.
<b>Field delimiter</b>	<p>(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.</p> <p>You can specify a non-printable character by entering the ASCII equivalent, such as:</p> <p>/ASCII_number</p>
<b>Maximum bind array</b>	<p>(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.</p> <p>If you do not enter a value, the default Oracle Bulk Loader value is used.</p>

Option	Description
<b>Use the control file</b>	<p>(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.</p> <p>Enter the name of the control file. Do not include the <code>.ctl</code> extension in the file name.</p> <p>If you do not specify a complete path, Data Services searches for the file in:</p> <ul style="list-style-type: none"> <li>• The path you have specified as the bulk loader directory in the datastore definition</li> <li>• <code>\\$LINK_DIR\log\bulkloader</code></li> </ul> <p>Both a control file and an associated data file must be in the same directory.</p> <p>If you select this option, you can also select these options or specify these values:</p> <ul style="list-style-type: none"> <li>• <b>Direct path</b></li> <li>• <b>Clean up bulk loader directory after load</b></li> <li>• <b>Rows per commit</b></li> <li>• <b>Maximum rejects</b></li> <li>• <b>Maximum bind array</b></li> </ul> <p>A variable can also be used.</p>
<b>Generate files only</b>	<p>(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.</p> <p>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 <code>tablename.ctl</code> and <code>tablename.dat</code>, where <code>tablename</code> is the name of the target table.</p>
<b>Direct path</b>	<p>(File method only) Select this check box to specify a direct-path load.</p> <p>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.</p>

Option	Description
<b>Clean up bulk loader directory after load</b>	(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 <code>.bad</code> file and does not delete any files. Errors occur when: <ul style="list-style-type: none"> <li>Log file was not created</li> <li>Log file contains “ORA-” or “SQL*Loader-”</li> </ul>
<b>Trailing nullcols</b>	(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.

### 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 2-164: Options

Option	Description
Table type	For template tables, select the appropriate table type for your SAP HANA target:  <b>Column Store</b> (default)  <b>Row Store</b>

Table 2-165: Bulk loading

Option	Description
Bulk load	Select to enable bulk loading.
Mode	Specify the mode for loading data to the target table: <b>Append:</b> Adds new records to the table (default). <b>Truncate:</b> Deletes all existing records in the table and then adds new records.

Option	Description
Commit size	<p>The value <b>default</b> 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):</p> <p><b>Column Store:</b> commit size is 10,000</p> <p><b>Row Store:</b> commit size is 1,000</p> <p>You can also type a custom value in the field (any value greater than 1).</p>
Update method	<p>Specify how the input rows are applied to the target table:</p> <p><b>Default:</b> Select to let Data Services apply the default value for this option based on the SAP HANA target table type:</p> <ul style="list-style-type: none"> <li>• <b>Column Store</b> tables use UPDATE.</li> <li>• <b>Row Store</b> tables use DELETE-INSERT.</li> </ul> <p><b>UPDATE:</b> Issues an UPDATE to the target table.</p> <p><b>DELETE-INSERT:</b> 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.</p> <p><b>Note:</b></p> <p>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.</p>

### Related Topics

- [Common target table options](#)
- [Performance Optimization Guide: Using Bulk Loading, Bulk loading in SAP HANA](#)

#### 2.2.23.3.11 Sybase ASE target table options

The following table contains option and description information specific to Sybase ASE target tables. All other option information for target tables can be found in the common table options (see related links below).

Table 2-166: Bulk loader options

Option	Description
Mode	Specify the mode for loading data in the target table: <b>Append:</b> Adds new records to the table. <b>Truncate:</b> Deletes all existing records in the table and then adds new records.
Bulk load	Select this check box to use 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.  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 Topics**

- [Common target table options](#)

**2.2.23.3.12 Sybase IQ target table options**

The following table contains option and description information specific to Sybase IQ target tables.

Table 2-167: Bulk loader options

Option	Description
Bulk load	Select this check box to use Sybase IQ bulk loading options to write the data.
Sybase IQ checkpoint	If selected, Data Services enables the 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.

Option	Description
Binary format	<p>Select the check box to load the staging and target tables in binary format. This format generally provides faster performance.</p> <p>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.</p>
Ignore conversion error	<p>Select to set the Sybase IQ database option <code>CONVERSION_ERROR</code> to OFF.</p> <p>If cleared, Data Services executes with the current database value for the option. For more information, see the Sybase IQ documentation.</p> <p>Conversion errors are reported as warnings in Data Services.</p>
Mode	<p>Specify the mode for loading data into the target table:</p> <p><b>Append:</b> Adds new records to the table.</p> <p><b>Truncate:</b> Deletes all existing records in the table and then adds new records.</p>
Block size (bytes)	The number of bytes read per block. Inappropriate adjustments to this option can dramatically affect performance. Defaults to 500,000 bytes.
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	<p>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.</p> <p>If cleared (the default), the software generates the data files and executes the bulk load.</p>
Clean up bulk loader directory after load	<p>If checked (the default), the software deletes the data file and auxiliary files after successfully completing the load.</p> <p>If cleared or if the load does not complete successfully, the data file and auxiliary files remain in the bulk loader directory.</p>
Error Handling	



Option	Description
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 Topics

- [Common target table options](#)

#### 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](#).

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 2-168: Common Teradata bulk loader options

Option	Description
File option	<p>Choose the type of file that will contain the data to bulk load:</p> <ul style="list-style-type: none"> <li>• Data File</li> <li>• Generic Named Pipe</li> <li>• Named Pipes Access Module</li> </ul>
<b>Data Services options</b>	
Generate files only	<p>When selected, the software generates a data file and a script file and ignores the <b>Number of loaders</b> option (on the <b>Options</b> 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.</p> <p>The software writes the data and control files in the bulk loader directory (default value is <code>&lt;DS_COMMON_DIR&gt;\log\bulkloader</code>) specified in the datastore definition. You must copy the files to the remote system manually. The naming conventions for these files is: <code>&lt;DatastoreName_OwnerName_TableName_n&gt;.ctl</code> and <code>&lt;DatastoreName_OwnerName_TableName_n&gt;.dat</code> where:</p> <ul style="list-style-type: none"> <li>• <i>OwnerName</i> is the table owner</li> <li>• <i>TableName</i> is the target table</li> <li>• <i>n</i> is a positive integer, optionally used to guarantee that the software does not overwrite a pre-existing file</li> </ul>
Clean up bulk loader directory after load	<p>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:</p> <ul style="list-style-type: none"> <li>• There is a syntax error in the script.</li> <li>• Error tables are not empty. Error tables contain rows that cannot be inserted into the target table due to data conversion or constraint violation.</li> </ul>
Mode	<p>Specifies the mode for loading data into the target table:</p> <p><b>Append:</b> Adds new records to the table.</p> <p><b>Replace:</b> Deletes all existing records in the table, and then inserts the loaded data as new records.</p>
Field delimiter	<p>Specifies a single-character field delimiter. Default value is /127 (non-printable character).</p>

Option	Description
Bulk operation	<p>For MultiLoad and TPump and for the Parallel Transporter bulk loader Update and Stream operators, specify the bulk operation to use:</p> <p><b>Insert:</b> Insert rows.</p> <p><b>Upsert :</b> If the row exists, update it; if not, insert it.</p>
<b>Named pipes access module</b>	For bulk loaders FastLoad, MultiLoad, TPump, and Parallel Transporter with the "File Option" <b>Named pipes access module</b> selected, the following "Named pipe parameters" are available.
Named pipe parameters	<p>You can override the default settings for the following Teradata Access Module parameters:</p> <ul style="list-style-type: none"> <li>• Logdirectory</li> <li>• Loglevel</li> <li>• Blocksize</li> <li>• FallbackFilename</li> <li>• FallbackDirectory</li> <li>• SignatureChecking</li> </ul> <p>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 <b>Log directory</b> 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.</p> <p>The bulk loader directory is the default value for both Logdirectory and FallbackDirectory.</p> <p>For more information about these Access Module parameters, see your Teradata tools and utilities documentation.</p>

Table 2-169: 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.

FastLoad paramaters	Description
Buffer size	<p>Output buffer size, in kilobyte,s used for Teradata FastLoad messages to the Teradata database.</p> <p>The output buffer size and the size of the rows in the Teradata FastLoad table determine the maximum 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.</p> <p>The default buffer size is also the maximum size allowed: 63 kb. If a value greater than 63 kb is specified, Teradata FastLoad:</p> <ul style="list-style-type: none"> <li>• Responds with a warning message</li> <li>• Resets the buffer size back to the default value</li> <li>• Continues with the Teradata FastLoad job</li> </ul>
Character set	Character set specification for the target.

Table 2-170: 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 2-171: 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.

TPump parameters	Description
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 workflow, 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 2-172: Parallel Transporter bulk loader

Option	Description
Operator	<p>Parallel Transporter operator values include:</p> <ul style="list-style-type: none"> <li>• Load</li> <li>• Update</li> <li>• Stream</li> <li>• SQL Inserter</li> </ul> <p>Note that the "Attributes" vary depending on the "Operator" selected. Refer to your Teradata documentation for information on attributes.</p>
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 <b>Data File</b> , 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.
<b>tbuild parameters</b>	
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.

Option	Description
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 to 10 seconds. Corresponds to tbuild -z option.

Table 2-173: Load Utilities

Option	Description
Command line	<p>Use this field to call a custom script. Load utilities include FastLoad, MultiLoad, and TPump. For example for FastLoad, you could enter:</p> <pre>fastload &lt; C:\Teradata\FLScripts\myScript.ctl</pre> <p>Data Services does not parse or modify these scripts.</p>
Named pipe name	<p>For a Load Utility, if you choose either <b>Named Pipes Access Module</b> or <b>Generic Named Pipes</b> file option, enter the pipe name.</p> <p>On UNIX, the pipe is a FIFO (first in, first out) file that has name of this format: /temp/<i>filename</i>.dat On Windows, the pipe name has this format: \\.\pipe\datastorename_ownername_tablename_loadernum.dat</p>

**Related Topics**

- [Common target table options](#)
- [Teradata](#)
- [Teradata source](#)
- [Performance Optimization Guide: Bulk Loading and Reading, Bulk loading and reading in Teradata](#)

**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 BusinessObjects Data Services uses to process large amounts of data.

**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	<p>(File targets only)</p> <p>The location relative to the Job Server of a file to use as the target. If the file does not exist, SAPBusinessObjects Data Services creates it. If the file exists, the software clears the content of the file before writing the output to it.</p> <p><b>Note:</b></p> <p>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. A variable can also be used.</p>
XML test file	<p>(Message targets only)</p> <p>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.</p> <p><b>Note:</b></p> <p>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. A variable can also be used.</p>
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 <b>With</b> field.

Option	Description
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.
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 Topics

- [Designer Guide: Embedded data flows](#)
- [XML template](#)
- [Locales and Multi-byte Functionality](#)

### 2.2.24 Target Writer migrated from Data Quality



If you migrate a Data Quality repository to SAP BusinessObjects Data Services and you have projects that contain database Writer transforms, the resulting SAP BusinessObjects 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*.

### 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, SAPBusinessObjects 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 2-175: Target tab

Option	Description
Table name	The name of the table. Can contain alpha, numeric, and underscores; cannot include blank spaces.

Table 2-176: Options tab

Option	Description
Column comparison	Drops the existing table and creates a new one with the same name before loading.
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 <b>Delete data from table before loading</b> option. These options are selected by default when you create a template table.

Option	Description
Use NVARCHAR for VARCHAR columns in supported databases	<p>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.</p> <p>If you are using an ODBC datastore to connect to Oracle, in the datastore editor for <b>ODBC Miscellaneous &gt; NVARCHAR type name</b>, select <b>NVARCHAR2</b>. See also <a href="#">ODBC</a>.</p> <p>The following database management systems do not support the nvarchar data type:</p> <ul style="list-style-type: none"> <li>• DB2 (non-UTF-8)</li> <li>• Oracle 8.x</li> <li>• Informix</li> <li>• Sybase ASE</li> <li>• Sybase IQ</li> </ul> <p>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.</p> <p><b>Caution:</b> 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.</p> <p><b>Caution:</b> Data truncation occurs when the column size of the source exceeds the maximum size allowed by the target DBMS.</p>

Table 2-177: Bulk Loader Options tab

Option	Description
Bulk load	Not available for template tables.

Table 2-178: 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 Topics

- [Target](#)
- [Message processing](#)

## 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.
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 Topics

- [Descriptions of transforms](#)

## 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 BusinessObjects 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 Topics

- [Catch](#)

## 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 Topics**

- [Designer Guide: While loops](#)
- [Smart editor](#)

## 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 BusinessObjects 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.



## 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 BusinessObjects 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.

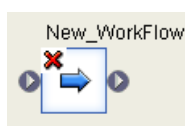
Subsequent flows wait until the software processes the first one. The engine provides a wait message for each subsequent work flow. Since only one Execute only once work flow can execute in a single job, the engine skips subsequent work flows and generates a second trace message for each, "Work flow 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.



## 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 BusinessObjects 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.

### 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 BusinessObjects Data Services prompts you to specify that the resulting object is a source or target file or a source or target message.

### 2.2.30.2 Parsing input and producing output

Mapping rules govern how SAP BusinessObjects Data Services translates an XML Schema or DTD into its internal schema definition and produces XML formatted data from an internal data set.

#### **Related Topics**

- [Designer Guide: Formatting XML documents](#)

### 2.2.30.3 Properties

XML file properties are the same as those of its format.

#### **Related Topics**

- [DTD](#)
- [XML schema](#)

## 2.2.31 XML message



XML  
Source



XML  
Target

### 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 BusinessObjects 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 Topics

- [DTD](#)
- [XML schema](#)

### 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.

### 2.2.31.2 Source and Target editors

You can find information about source and target options elsewhere in the *Reference Guide*.

#### Related Topics

- [XML message source](#)
- [Target XML files, messages, and templates](#)

### 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.

### 2.2.31.4 Parsing input and producing output

Mapping rules govern how SAP BusinessObjects 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 Topics

- [DTD](#)
- [XML schema](#)

### 2.2.31.5 Properties

XML file properties are the same as those of its format.

### Related Topics

- [DTD](#)
- [XML schema](#)

## 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 BusinessObjects 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/](http://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 Topics

- [Rules for importing XML Schemas](#)

### 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: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>

```

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 Topics

- [Designer Guide: Formatting XML documents](#)

## 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.

XML schema property	Description
Imported from	The full path to the format. For example, D:\data\test.xsd.
Root element name	The name of the primary node you want to import. SAP BusinessObjects 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.

### 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 was an element or attribute in the original XML Schema.
Length	Number of characters in this column.

Column attribute	Description
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 BusinessObjects 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. 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 Topics

- [Column attributes for tables](#)

## 2.2.32.4 Supported XML Schema components

SAP BusinessObjects 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 Topics

- [Unsupported XML schemas](#)

## 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.

### 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 BusinessObjects 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".

## 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.



XML Schema Element	Attribute (nested table or column)
Choice	<p>Any One Column.</p> <p>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.</p> <p>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.</p> <p><i>Sequence</i> , <i>choice</i> , and <i>all</i> are handled as follows:</p> <ul style="list-style-type: none"> <li>Sequence becomes "Any One Column = NO". Attributes A, B, and C become columns A, B, and C.</li> <li>Choice becomes "Any One Column = YES": Attributes A, B, and C become columns A or B or C.</li> <li>All becomes "All": B, C, and A or any combination of the three.</li> </ul>
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
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 <code>varchar</code> ).
Sequence	See Choice.

### Related Topics

- [Unsupported XML schemas](#)

## 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 <code>varchar</code> ).
Use	An XML Schema <i>Use</i> attribute with a value of OPTIONAL becomes the <i>Required</i> attribute with a value of NO.  An XML Schema <i>Use</i> attribute with a value of REQUIRED becomes the <i>Required</i> attribute with a value of YES.

### Related Topics

- [Unsupported XML schemas](#)

## 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 BusinessObjects 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.

## 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 BusinessObjects Data Services any reference to a group will be replaced by the contents of that group.

## 2.2.32.10 Rules for importing XML Schemas

SAP BusinessObjects 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.

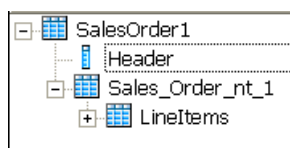
After applying these rules, the software uses two additional rules, except where doing so would allow more than one row for a root element:

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.

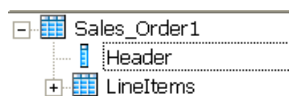
For example, the SalesOrder element might be defined as follows in an XML Schema:

```
<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.

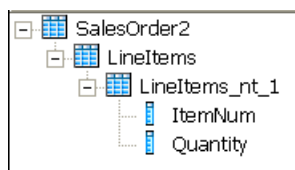
For example, the nested table LineItems might be defined as follows in an XML Schema:

```

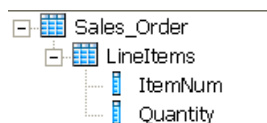
<xs:element name="SalesOrder">
  <xs:element name="LineItems" minOccurs="0"
    maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ItemNum"/>
        <xs:element ref="Quantity"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

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.



## 2.2.32.11 Unsupported XML schemas

The following XML Schema elements and attributes are not supported in SAP BusinessObjects Data Services. They are ignored and not imported.

Component	Description
Annotation	The software 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.

## 2.2.32.12 Limitations

If an XML schema definition contains the following elements or attributes, SAP BusinessObjects 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.

### 2.2.32.13 Data type mappings

SAP BusinessObjects 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.

### 2.2.32.14 Primitive types

The table below lists Primitive XML Schema types, examples, and the corresponding data type in SAP BusinessObjects Data Services. The constraining facets used are shown in bold.

XML Schema type	Example	Data type
AnyURI	<code>http://www.example.com/</code>	<code>Varchar(len) : len = <b>length</b> (in chars)</code>
Base64Binary	<code>GpM7</code>	<code>Varchar(len) : len = <b>length</b> (in octets)</code>
Boolean	<code>{true, false, 0, 1}</code>	<code>Varchar(5)</code>
Date	<code>CCYY-MM-DD</code>	<code>Datetime</code>
DateTime	<code>Format = CCYY-MM-DD HH:MM:SS</code>	<code>Datetime</code>
Decimal	<code>7.999</code>	<code>Decimal(p, s) : p = <b>totalDigits</b>, a maximum of 28 and s = <b>fractionDigits</b>, default =28,2</code>
Double	<code>64 bit floating point</code>	<code>Double (In the software there is no difference between real and double)</code>
Duration	<code>P1Y2M3DT10H30M</code>	<code>Varchar(64)</code>

XML Schema type	Example	Data type
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 = <b>length</b> (in octets)
Notation		N/A
Qname	po:USAddress	Varchar(len) : len = <b>length</b> (in chars)
String	"Hello World"	Varchar(len) : len = <b>length</b> (in characters)
Time	HH:MM:SS	Datetime

### 2.2.32.15 Derived types

The table below lists pre-defined Derived XML Schema types, examples, and the corresponding data type in SAP BusinessObjects 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 = <b>length</b> (in characters)
Token		Varchar(len) : len = <b>length</b> (in characters)
Language	En-GB, en-US, fr	Varchar(len): len = <b>length</b> (in characters)
NMTOKEN	US, Brésil	Varchar(len): len = <b>length</b> (in characters)

XML Schema type	Example	Data type
NMTOKENS	Brésil Canada Mexique	Varchar(len): len = <b>length</b> (in characters)
Name	ShipTo	Varchar(len): len = <b>length</b> (in characters)
NCName	USAddress	Varchar(len): len = <b>length</b> (in characters)
ID		Varchar(len): len = <b>length</b> (in characters)
IDREF		Varchar(len): len = <b>length</b> (in characters)
IDREFS		Varchar(len): len = <b>length</b> (in characters)
ENTITY		Varchar(len): len = <b>length</b> (in characters)
ENTITIES		Varchar(len): len = <b>length</b> (in characters)
Integer		Int
NonPositiveInteger		Int
NegativeInteger		Int
Long		Decimal 28,0
Int		Int
Short		Int
Byte		Int
NonNegativeInteger		Int
UnsignedLong		Long
UnsignedShort		Int
UnsignedByte		Int
PositiveInteger		Int
AnyType (ur-type)	unspecified type	Varchar(255)



### 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 BusinessObjects 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.

### 2.2.32.17 List types

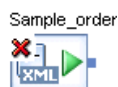
XML Schemas have list types. When it encounters a list, SAP BusinessObjects Data Services makes that list's corresponding data type a `varchar(1024)`. All the elements of the list are placed in the value of that column as a string (exactly as it is represented in the XML).

### 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 BusinessObjects Data Services makes that union's corresponding data type a `varchar (1024)`.

### 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 BusinessObjects 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.

### 2.2.32.20 Error checking

SAP BusinessObjects 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", "StainlessSteel", 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.

### 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.

### 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 Topics

- [Target XML files, messages, and templates](#)



## 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 Topics

- [Scripting Language](#)
- [Custom functions](#)

### 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.

#### 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

## 3.2 Smart editor options

### 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.
Validate	Click to check your script for errors.

## 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.

### 3.2.2.1 Tabs

The **Functions** tab displays existing functions in SAP BusinessObjects 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.

### 3.2.2.2 Find option

You can search or browse through each tab to find the content you want to include in your script or expression.

## 3.2.3 Editor pane

### 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

### 3.2.3.2 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.

#### 3.2.3.2.1 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.

### 3.2.3.3 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
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.

### 3.2.3.4 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.

#### 3.2.3.4.1 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.

#### 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 Topics

- [Debugging and Validation](#)

## 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.

### 3.4 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**



# 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 BusinessObjects Data Services processes, converts, and evaluates data types.

## 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 BusinessObjects Data Services:

- Data read from sources is converted to the appropriate SAP BusinessObjects Data Services data types.
- Data loaded to targets is converted from their SAP BusinessObjects Data Services data types to types appropriate for the target.

The software recognizes the following data types:

- [date](#)
- [datetime](#)
- [decimal](#)
- [double](#)
- [int \(integer\)](#)
- [interval](#)
- [long](#)
- [numeric](#)
- [real](#)
- [time](#)
- [timestamp](#)
- [varchar](#)

All of these data types allow NULL values.

### 4.1.1 date

The date data type defines calendar dates.

SAP BusinessObjects 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

Date format code	Description	Example
YY	<p>2-digit year</p> <p>A YY less than 15 is interpreted as being 20 YY ; for example, 10 would be interpreted as being the year 2010.</p> <p>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.</p> <p>To change the value that the software uses to interpret 2-digit year dates, change the <b>Century Change Year</b> value in the Data options. (Select <b>Tools &gt; Options</b> to open the "Options" window, and then select <b>General</b> under the <b>Data</b> category). The value must be a positive integer between 0 and 99.</p>	<p>The year 1998:</p> <p>98</p>
YYYY	4-digit year	<p>The year 1999:</p> <p>1999</p>

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
<code>to_char(\$MyDate, 'YYYY.MM.DD')</code>	1996.01.01
<code>to_char(\$MyDate, 'MONTH DD, YYYY')</code>	JANUARY 01, 1996
<code>to_char(\$MyDate, 'DD/MM/YY')</code>	01/01/96
<code>to_date('01/01/96', 'DD/MM/YY')</code>	1996.01.01 stored as a date

Example	Output
<code>to_date('01/01/19', 'DD/MM/YY')</code>	2019.01.01 stored as a date

**Related Topics**

- [Date arithmetic](#)
- [Platform support for NULL values](#)

## 4.1.2 datetime

The datetime data type defines calendar dates and times.

SAP BusinessObjects 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 BusinessObjects 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 Topics**

- [Date arithmetic](#)

## 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:

$1 \leq \text{precision} \leq 96$

$0 \leq \text{scale} \leq \text{precision}$

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.

**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.

**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 Topics**

- [Oracle](#)

## 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 BusinessObjects Data Services is running.

## 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.

### 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 BusinessObjects 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; see [Date arithmetic](#).

This data type allows NULL values.

### 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

Transform	Do not use long or blob column data in
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 BusinessObjects 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.

**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 Topics**

- [Informix](#)

### 4.1.7.1 long

SAP BusinessObjects 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 SAP BusinessObjects 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 SAP BusinessObjects 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
Description	long

Column name	Data type
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 Topics

- [Limitations for long and blob](#)
- [blob](#)
- [Conversion to or from internal data types](#)

## 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 BusinessObjects 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**

SAP BusinessObjects 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
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 Topics**

- [long](#)
- [Conversion to or from internal data types](#)
- [Limitations for long and blob](#)

## 4.1.8 numeric

The `decimal` data type and the `numeric` data type are identical in SAP BusinessObjects Data Services. See [decimal](#) for data type information.

## 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 BusinessObjects 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.

#### 4.1.10 time

The time data type defines times of the day, with no calendar date.

SAP BusinessObjects 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
<code>to_char(\$MyTime, 'HH24:MI:SS.FF')</code>	20:25:00
<code>to_char(\$MyTime, 'HH24:MI')</code>	20:25

You can add and subtract date, datetime, interval, and time values.

### Related Topics

- [Date arithmetic](#)

## 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](#) 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 BusinessObjects 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 BusinessObjects 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.

#### 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 BusinessObjects 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 BusinessObjects 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 SAP BusinessObjects 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 Topics

- [NULL values and empty strings](#)
- [Processing with and without UTF-16 Unicode](#)

## 4.2 Data type conversion

This section discusses how Data Services processes various data types—conversions during arithmetic operations and between data types.

### 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
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

### 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.

### 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.

**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.

### 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 <b>Note:</b> 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

Attunity Streams data type	Converts to Data Services data type
integer	int
smallint	int
time	time
timestamp	datetime
varchar	varchar

#### 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
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

COBOL copybook data type	Converts to Data Services data type
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

#### 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
smallint	int
int	int
bigint	decimal(20,0)
float	real
double	double

Hadoop Hive data type	Converts to Data Services data type
string	varchar
boolean	varchar(5)
complex	not supported

#### 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 $\leq 18$ Numeric: if precision is $> 18$
real	float
time	time
varchar	varchar
timestamp	timestamp

#### 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 <b>Note:</b> Because int is only four bytes, data is lost during the conversion to Data Services data type.
blob	blob
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

Data Services data type	DB2 data type in template table
int	int
interval	real
long	clob
numeric	character
real	real
time	time
varchar	varchar
timestamp	date

#### 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

<b>Informix data type</b>	<b>Converts to or from Data Services data type</b>
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.

<b>Data Services data type</b>	<b>Informix data type in template table</b>
blob	blob
date	date
datetime	date
decimal	decimal
double	float
int	int
interval	int
long	clob
numeric	decimal
real	real
time	date
varchar	varchar
timestamp	date

#### **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

#### 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
datetime	datetime
datetime2 (SQL Server 2008 and higher only)	datetime
decimal	decimal
float	double
image	blob
int	int
money/smallmoney	decimal
nchar	varchar

Microsoft SQL Server data type	Converts to or from Data Services data type
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
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 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

Data Services data type	MS SQL Server data type in template table
numeric	decimal
real	real
time	datetime time (SQL Server 2008 and higher only)
varchar	varchar
timestamp	datetime datetime2 (SQL Server 2008 and higher only)

#### 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
float	real
double	double
datetime	datetime

MySQL data type	Converts to or from Data Services data type
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

Data Services data type	MySQL data type in template table
time	time
varchar	varchar
timestamp	timestamp

#### 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



Netezza data type	Converts to or from Data Services data type
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
int	integer
interval	real
numeric	numeric
real	real
time	time
timestamp	timestamp
varchar	varchar

#### 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

ODBC data type	Converts to or from Data Services data type
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 from internal 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

Data Services data type	ODBC data type in template table
time	time
varchar	varchar
timestamp	timestamp

#### 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 BusinessObjects 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 decimal: All other
nvarchar2	varchar
real	double
row	not supported
rowid	not supported

Oracle data type	Converts to or from SAP BusinessObjects Data Services data type
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

#### 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.

<b>SAP HANA data type</b>	<b>Converts to Data Services data type</b>
integer	int
tinyint	int
smallint	int
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.

<b>Data Services data type</b>	<b>Converts to SAP HANA data type</b>
blob	blob
date	date
datetime	timestamp
decimal	decimal
double	double
int	integer
interval	real

Data Services data type	Converts to SAP HANA data type
long	clob/nclob
real	decimal
time	time
timestamp	timestamp
varchar	varchar/nvarchar

#### 4.2.2.15 Sybase ASE

The following table shows the conversion between 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.

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

Sybase ASE data type	Converts to or from Data Services data type
tinyint	int
varbinary	not supported
varchar	varchar

The following table shows the conversion from internal data types to Sybase ASE data types in template tables or Data\_Transfer transform tables.

Data Services data type	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
varchar	varchar
timestamp	datetime

#### 4.2.2.16 Sybase IQ

The following table shows the conversion between 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.

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
unsigned int	int
varbinary	not supported
varchar	varchar

The following table shows the conversion from internal data types to Sybase IQ data types in template tables or Data\_Transfer transform tables.



Data Services data type	Sybase IQ data type in template table
blob	longbinary
date	date
datetime	timestamp
decimal	decimal
double	double
int	int
interval	int
long	N/A
numeric	N/A
real	real
time	time
varchar	varchar
timestamp	timestamp

#### 4.2.2.17 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
float	float

Teradata data types	Converts to or from Data Services data type
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

Data Services data type	Teradata data type in template table
time	time
varchar	varchar
timestamp	N/A

### 4.2.3 Conversion of data types within expressions

When possible, SAP BusinessObjects 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](#)).

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.

### 4.2.4 Conversion among number data types

SAP BusinessObjects 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)
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 4-12: 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 4-13: 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
String	String to number	String to datetime	String to real	String to number
Interval	Number to interval	Illegal	String to interval	OK

Table 4-14: 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 BusinessObjects 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.

## 4.2.5 Conversion between explicit data types

You can use functions to convert data from one type to another:

- [interval\\_to\\_char](#)
- [julian\\_to\\_date](#)
- [num\\_to\\_interval](#)
- [to\\_char](#)
- [to\\_date](#)
- [to\\_decimal](#)

You can also import database-specific functions to perform data type conversions.

## 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.

### 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

### 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

### 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

### 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



#### 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

#### 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.



# 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.

## 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	<p>Creates a new row in the target.</p> <p>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.</p> <p>Most transforms operate only on rows flagged as NORMAL.</p>
INSERT	<p>Creates a new row in the target.</p> <p>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.</p> <p>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.</p>
DELETE	<p>Is ignored by the target. Rows flagged as DELETE are not loaded.</p> <p>Rows can be flagged as DELETE in the Map_Operation and Table Comparison transforms. Only the History_Preserving, transform with the <b>Preserve delete row(s) as update row(s)</b> option selected, can accept data sets with rows flagged as DELETE.</p>

Operation Code	Description
UPDATE	<p>Overwrites an existing row in the target table.</p> <p>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.</p> <p>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.</p>

## 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 5-2: 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.

Transform	Description
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 5-3: 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.
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.

Transform	Description
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 5-4: 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	Allows conversions between operation codes.
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 5-5: 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.

**Note:**

For all transforms, to refresh a target schema after making changes to transform options, choose **View** > **Refresh** or press **F5**.

**Related Topics**

- [Associate](#)
- [Case](#)
- [Country ID](#)
- [Data Cleanse](#)
- [Data\\_Transfer](#)
- [Date\\_Generation](#)
- [Effective\\_Date](#)
- [Entity Extraction transform](#)
- [Geocoder](#)
- [Global Address Cleanse](#)
- [Global Suggestion List](#)
- [Hierarchy\\_Flattening](#)
- [History\\_Preserving](#)
- [Key\\_Generation](#)
- [Map\\_CDC\\_Operation](#)
- [Map\\_Operation](#)
- [Match](#)
- [Merge](#)
- [Pivot \(Columns to Rows\)](#)
- [Query](#)
- [Reverse Pivot \(Rows to Columns\)](#)
- [Row\\_Generation](#)
- [SQL](#)
- [Table\\_Comparison](#)
- [USA Regulatory Address Cleanse](#)
- [User-Defined](#)
- [Validation](#)
- [XML\\_Pipeline](#)
- [DSF2® Walk Sequencer](#)

## 5.3 Data Integrator transforms

### 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 Topics

- [Performance Optimization Guide: Splitting a data flow into sub data flows](#)
- [Target options](#)

#### 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.

#### 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.

### 5.3.1.3 Target options

The tabs that appears on the target editor depends on the transfer type that you specify.

#### 5.3.1.3.1 General tab

Transfer type	Option	Description
File, Table, or Automatic	<b>Enable transfer</b>	Enables or disables the execution of the <code>Data_Transfer</code> 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 <code>Data_Transfer</code> transform.  <b>Note:</b> When you run the job in debug mode, Data Services automatically disables all <code>Data_Transfer</code> transforms.
File	<b>File options: File name</b>	Name of the flat file that you want to use as transfer for sub data flows. The file does not need to exist.
File	<b>File options: Root directory</b>	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 <b>Root directory</b> . If your default Job Server does not reside on your local computer, you must manually enter the path to your <b>Root directory</b> .  You can use a global variable or parameter for the path-name.

Transfer type	Option	Description
File, Table, or Automatic	<b>Join rank</b>	<p>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.</p> <p>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.</p> <p>Must be a non-negative integer. Default value is 0.</p>
Table - relational	<b>Table options: Array fetch size</b>	<p>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.</p> <p>This option is available for source tables from DB2, Informix, ODBC, Oracle, and SQL Server datastores.</p> <p>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.</p>

Transfer type	Option	Description
Table - relational	<b>Table options: Database type</b>	<p>Select an item in the <b>Database type</b> 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.</p> <p>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.</p> <p>Data Services allows you to use target table editor option values from any datastore configuration:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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 <b>Use values from</b> box in the Create New Configuration dialog (a sub-dialog of the datastore editor).</li> <li>• If you also select the <b>Restore values if they already exist</b> 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 <b>Use values from</b> option.</li> </ul> <p>For example, suppose you set a configuration for Oracle 8i. When you edit the target table editor options, you change the <b>Rows Per Commit</b> 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 <b>Use values from</b> option to Oracle 8i. In this case, the target table editor settings for SQL Server inherit the value 500 for <b>Rows per Commit</b> because this was the value set in the Oracle 8i configuration.</p>

Transfer type	Option	Description
Table - relational	<b>Table options: Table name</b>	<p>Name of the database table that you want to use as transfer for sub data flows. Specify the table name with the following format:</p> <pre>datastorename.ownername.tablename</pre> <p>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.</p>
File, Table, or Automatic	<b>Transfer type</b>	<p>Choose one of the following transfer types to temporarily store the data of each sub data flow:</p> <ul style="list-style-type: none"> <li>• <b>Table:</b> Database table from an existing datastore. If you choose this type, specify the Table options below (<b>Table name</b>, <b>Database type</b>, and <b>Array fetch size</b>).</li> <li>• <b>File:</b> A flat file. If you choose this type, specify the File options below (<b>Root directory</b> and <b>File name</b>).</li> <li>• <b>Automatic:</b> The Data Services optimizer chooses the transfer type from: <ul style="list-style-type: none"> <li>• Your datastores that selected the <b>Enable automatic data transfer</b> check box, or</li> <li>• The pageable cache directory that you specify in the Server Manager.</li> </ul> </li> </ul> <p>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.</p> <p>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.</p> <p>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 each job server in a server group), the optimizer chooses the directory that is local to the data flow process.</p>

## 5.3.1.3.2 Options tab

Transfer type	Option	Description
Table - relational database	<b>Data definition language (DDL)</b>	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	<b>Delete data before loading</b>	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	<b>Drop and re-create before loading</b>	Drops the existing table and creates a new one with the same name before loading. Defaults to selected.  <b>Note:</b> Unlike a template table, you can use bulk loading options for a transfer table even when the <b>Drop and re-create before loading</b> is checked.
Table - relational database	<b>Enable partitioning</b>	(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 <b>Enable partitioning</b> , Data Services transfers data using the number of partitions in the table as the maximum number of parallel instances.  If you select <b>Enable partitioning</b> , you cannot select <b>Number of Loaders</b> .
Table - relational database	<b>Generate default DDL</b>	Click this button to display the SQL CREATE TABLE statement that Data Services generates.

Transfer type	Option	Description
Table - relational database	<b>Number of loaders</b>	<p>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.</p> <p>If you select <b>Number of Loaders</b>, you cannot select <b>Enable partitioning</b>.</p> <p>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.</p> <p>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.</p>
Table - relational database	<b>Rows per commit</b>	<p>Specifies the transaction size in number of rows.</p> <p>If set to 1000, Data Services sends a commit to the underlying database every 1000 rows.</p> <p>This option is not available for targets in real time jobs.</p>

#### 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 <a href="#">Target tables</a> .

#### 5.3.1.3.4 Pre-Load Commands and Post-Load Commands tabs

Transfer type	Option	Description
Table - relational database	-	

Transfer type	Option	Description																					
		<p>Specify SQL commands that Data Services executes before starting a load or after finishing a load into a transfer table.</p> <p>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.</p> <p><b>Note:</b> Because Data Services executes the SQL commands as a unit of transaction, you should not include transaction commands in preload or postload SQL statements.</p> <p>Both the <b>Pre Load Commands</b> tab and the <b>Post Load Commands</b> tab contain a <b>SQL Commands</b> box and a <b>Value</b> box. The <b>SQL Commands</b> box contains command lines. When you first open the tab, an empty line appears.</p> <p>To edit a line, select the line in the <b>SQL Commands</b> box. The text for the SQL command appears in the <b>Value</b> box. Edit the text in that box.</p> <p>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 <b>Insert Before</b> to insert a new line before the selected line, or choose <b>Insert After</b> to insert a new line after the selected line. Finally, type the SQL command in the <b>Value</b> box.</p> <p>To delete a line, select the line in the <b>SQL Commands</b> box, right click, and choose <b>Delete</b>.</p> <p>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.</p> <table border="1"> <thead> <tr> <th>Entered statement</th><th>Variable value</th><th>Written statement</th></tr> </thead> <tbody> <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> </tbody> </table> <p>You cannot use Pre Load and Post Load SQL commands in a real-time job.</p>	Entered statement	Variable value	Written statement	[\$X]	5	5	[\$X]	John Smith	John Smith	{ \$X }	5	5	{ \$X }	John Smith	John Smith	'\$X'	5	5	'\$X'	John Smith	John Smith
Entered statement	Variable value	Written statement																					
[\$X]	5	5																					
[\$X]	John Smith	John Smith																					
{ \$X }	5	5																					
{ \$X }	John Smith	John Smith																					
'\$X'	5	5																					
'\$X'	John Smith	John Smith																					



### 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
```

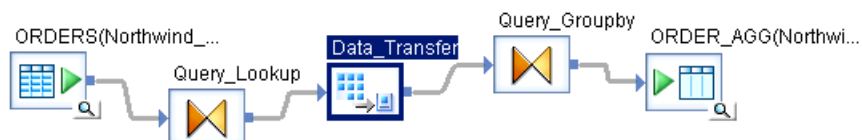
### 5.3.1.5 Example

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.
  
```

## 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.

### 5.3.2.1 Data inputs

None.

### 5.3.2.2 Options

Option	Description
Cache	<p>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.</p> <p>Options are:</p> <ul style="list-style-type: none"><li>• <b>Yes</b>: The source is always cached unless it is the outer-most source in a join.</li><li>• <b>No</b>: The source is never cached.</li></ul> <p>The default is <b>Yes</b>.</p> <p>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.</p>

Option	Description
<b>End date</b>	The last date in the output sequence. Use the same format used for <b>Start date</b> to specify this date.
<b>Increment</b>	The interval between dates in the output sequence. Select <b>Daily</b> , <b>Monthly</b> , or <b>Weekly</b> .
<b>Join rank</b>	<p>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.</p> <p>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.</p> <p>Must be a non-negative integer. Default value is 0.</p>
<b>Start date</b>	<p>The first date in the output sequence. Specify this date using the following format:</p> <p>YYYY.MM.DD</p> <p>where YYYY is a year value, MM is a month value, and DD is a day value.</p>

### 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.

### 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)`

### 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.

#### 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 Reser- voir	1999.08.02	0	Proposal
Hetch Hetchy Reser- voir	2002.10.17	0a	Proposal
Hetch Hetchy Reser- voir	2002.10.17	1a	Case
Hetch Hetchy Reser- voir	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.

### 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**.

### 5.3.3.3 Options

Option	Description
<b>Default effective to date value</b>	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.
<b>Effective date column</b>	<p>A column in the input data set of type <code>date</code> that contains the effective date. This column name is entered automatically if Data Services finds a column named <code>EFFDT</code> in the source. The column appears in the output schema with the name <code>EFFDT</code>.</p> <p>This field is required.</p>

Option	Description
<b>Effective sequence column</b>	<p>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.</p> <p>This field is required only if the input data set allows duplicate effective dates.</p>
<b>Effective to column</b>	<p>The name of a date column added to the output schema that contains the effective-to date.</p> <p>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 <b>Default effective to date</b> is used.</p>

### 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	2999.01.01a	Case
Hetch Hetchy Reservoir	1999.08.02	2002.10.17	Proposal
Hetch Hetchy Reservoir	2002.10.17	2003.05.06	Case



Project	Effective date	Effective-to date	Status
Hetch Hetchy Reservoir	2003.05.06	2999.01.01a	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	1a	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.

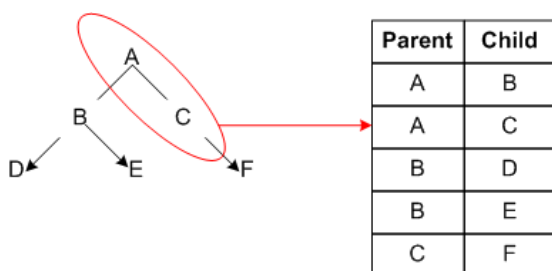
## 5.3.4 Hierarchy\_Flattening



Constructs a complete hierarchy from parent/child relationships, then produces a description of the hierarchy in vertically or horizontally flattened format.

### 5.3.4.1 Data inputs

Rows describing individual parent-child relationships.



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.

### 5.3.4.2 Editor

The Hierarchy\_Flattening editor includes:


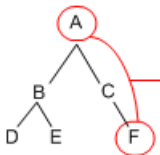
- Schema In pane which contains the source schema
- Schema Out pane which contains the target schema
- Hierarchy\_Flattening transform options

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.

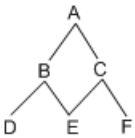

Data Services generates the target schema in response to the values you choose in the transform options. To refresh the target schema after you make a change to the options, choose **View > Refresh** or press **F5**.

### 5.3.4.3 Options

Option	Description									
Child attribute list	Identifies a column or columns that are associated with the child column. You can drag columns from the source schema into the <b>Child attributelist</b> . 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>	Flattening mode	Source column	Target column	Vertical	POPULATION	C_POPULATION	Horizontal	POPULATION	C_L1_POPULATION C_L2_POPULATION (one column for each hierarchy level)
	Flattening mode	Source column	Target column							
	Vertical	POPULATION	C_POPULATION							
Horizontal	POPULATION	C_L1_POPULATION C_L2_POPULATION (one column for each hierarchy level)								
You can specify columns including nested schemas as the child attribute.										
Child column	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 <b>Child column</b> box.  You cannot specify columns including nested schemas as the child.									
Do not abort in case of cycle	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.									

Option	Description																																																																																																							
Flattening type	<p>Indicates how the hierarchical relationships are described in the output. Choose from two options:</p> <ul style="list-style-type: none"><li><b>Horizontal</b> — 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 <b>Maximum depth</b>.</li></ul> <div><table><thead><tr><th>Node</th><th>Level0</th><th>Level1</th><th>Level2</th></tr></thead><tbody><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></tbody></table></div> <ul style="list-style-type: none"><li><b>Vertical</b> — 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 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.</li></ul> <div><table><thead><tr><th>Ancestor</th><th>Descendent</th><th>Depth</th><th>Root_flag</th><th>Leaf_flag</th></tr></thead><tbody><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></tbody></table></div>	Node	Level0	Level1	Level2	A	A			B	A	B		D	A	B	D	E	A	B	E	C	A	C		F	A	C	F	Ancestor	Descendent	Depth	Root_flag	Leaf_flag	B	D	1	0	1	B	E	1	0	1	B	B	0	0	0	A	B	1	1	0	A	D	2	1	1	A	E	2	1	1	A	C	1	1	0	A	F	2	1	1	A	A	0	1	0	D	D	0	0	1	C	F	1	0	1	C	C	0	0	0	E	E	0	0	1	F	F	0	0	1
	Node	Level0	Level1	Level2																																																																																																				
A	A																																																																																																							
B	A	B																																																																																																						
D	A	B	D																																																																																																					
E	A	B	E																																																																																																					
C	A	C																																																																																																						
F	A	C	F																																																																																																					
Ancestor	Descendent	Depth	Root_flag	Leaf_flag																																																																																																				
B	D	1	0	1																																																																																																				
B	E	1	0	1																																																																																																				
B	B	0	0	0																																																																																																				
A	B	1	1	0																																																																																																				
A	D	2	1	1																																																																																																				
A	E	2	1	1																																																																																																				
A	C	1	1	0																																																																																																				
A	F	2	1	1																																																																																																				
A	A	0	1	0																																																																																																				
D	D	0	0	1																																																																																																				
C	F	1	0	1																																																																																																				
C	C	0	0	0																																																																																																				
E	E	0	0	1																																																																																																				
F	F	0	0	1																																																																																																				
Generate cycle rows	<p>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).</p> <p>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.</p>																																																																																																							

Option	Description									
Maximum depth	<p>(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.</p> <p>If you do not know the number of levels in your hierarchy, set <b>Maximum depth</b> to 1. When you execute the job, a warning message will appear in the execution log indicating that the <b>Maximum depth</b> is less than the actual depth of the hierarchy. Reset <b>Maximum depth</b> to the actual value reported in the warning message.</p>									
Parent attribute list	<p>Identifies a column or columns that are associated with the parent column. You can drag columns from the source schema into the <b>Parent attributelist</b>. 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 <code>POPULATION</code> to the parent attribute list.</p> <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 (one column for each hierarchy level)</td></tr></table> <p>You can specify columns including nested schemas as the parent attribute.</p>	Flattening mode	Source column	Target column	Vertical	POPULATION	P_POPULATION	Horizontal	POPULATION	P_L1_POPULATION P_L2_POPULATION (one column for each hierarchy level)
Flattening mode	Source column	Target column								
Vertical	POPULATION	P_POPULATION								
Horizontal	POPULATION	P_L1_POPULATION P_L2_POPULATION (one column for each hierarchy level)								
Parent column	<p>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 <b>Parent column</b> box.</p> <p>You cannot specify columns including nested schemas as the parent.</p>									
Run as separate process	<p>This option creates a separate sub data flow process for the Hierarchy_Flattening transform when Data Services executes the data flow.</p>									

Option	Description
Use maximum length paths	<p>(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.</p> <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;">  <p>The depth of node E is the same for both paths to A</p> </div> <div style="text-align: center;">  <p>The depth of node M is "1" or "2" depending on the path to J</p> </div> </div>

### Related Topics

- [Performance Optimization Guide: Run as a separate process option](#)

## 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	<p>The number of levels down from the root node where Current_leaf is found</p> <p>The root node has Leaf_level of 0.</p>
LEVEL0	The descriptor for the top level node.

Column name	Description
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

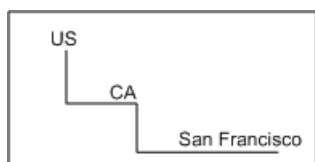
In a typical data flow including a Hierarchy\_Flattening transform with the same attribute for parent and child nodes, you may follow the transform with a query that would filter the duplicated attribute values. The query would pass the *P\_L0\_attribute\_column* and the *C\_Ln\_attribute\_column* through 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.

Column name	Description
<i>P_attribute_column</i>	Column from the source that you associate with the parent (can be more than one <i>P_attribute_column</i> ). If Leaf_flag is 1, this value is NULL.
<i>C_attribute_column</i>	Column from the source that you associate with the child (can be more than one <i>C_attribute_column</i> ). 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.

### 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.

**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.

## 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.

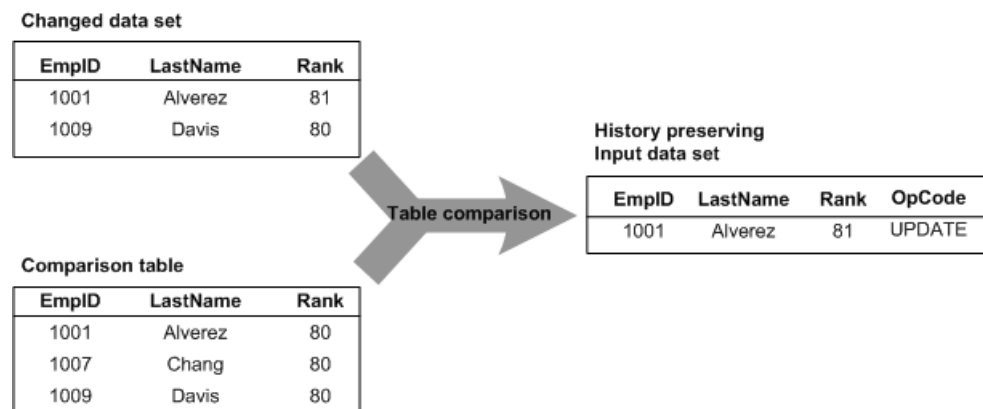
### Related Topics

- [Table\\_Comparison](#)
- [real](#)

### 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 Alvarez and discards unchanged data for employee Davis. The result is a single row flagged with the UPDATE operation code.



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.

### 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**.

### 5.3.5.3 Options

Option	Description
<b>Compare columns</b>	<p>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.</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• <b>Compare columns</b> cannot contain nested schemas.</li> </ul>
<b>Current flag—Column</b>	<p>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.</p> <p>The current flag <b>Column</b> cannot be the same value as the <b>Valid from</b> or the <b>Valid to</b> date column. The current flag <b>Column</b> cannot contain a nested schema. Data Services validates that you specify a current flag <b>Column</b>, <b>Set value</b>, and <b>Reset value</b>, if any are specified.</p>
<b>Current flag—Reset value</b>	<p>An expression that evaluates to a value with the same data type as current flag <b>Column</b>. Data Services uses this value to update the current flag <b>Column</b> 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 <b>Column</b> box to enable the <b>Flag reset value</b>. For added flexibility, you can enter a variable for this option.</p>
<b>Current flag—Set value</b>	<p>An expression that evaluates to a value with the same data type as the current flag <b>Column</b>. Data Services uses this value to update the current flag <b>Column</b> in the new row in the warehouse added to preserve history of an existing row. Enter a value in the Current flag <b>Column</b> box to enable the <b>Set value</b>. For added flexibility, you can enter a variable for this option.</p>
<b>Date columns—Valid from</b>	<p>A date or datetime column from the source schema. If the warehouse uses an effective date to track changes in data, specify a <b>Valid from</b> date column. Data Services 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 <b>Valid to</b> date column in the previously current row in the warehouse.</p> <p>Data Services validates that both the <b>Valid from</b> date column and the <b>Valid to</b> date column have been specified if either is specified.</p>

Option	Description																																				
Date columns—Valid to	<p>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 <b>Valid from</b> date column.</p> <p>This value is used as the new value in the <b>Valid to</b> date column in the new row added to the warehouse to preserve history of an existing row.</p> <p>The <b>Valid to</b> date column cannot be the same as the <b>Valid from</b> date column.</p>																																				
Date columns—Valid to date value	<p>Specify the values to use in the <b>Valid to</b> date column in the old record and the new record added to the warehouse to preserve history of an existing row.</p> <ul style="list-style-type: none"><li><b>New record</b>—You can specify one of the following values:<ul style="list-style-type: none"><li>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.</li><li>A variable name that contains a date value.</li></ul></li><li><b>Old record</b>—You can specify one of the following values:<ul style="list-style-type: none"><li><b>Use "valid from" date of new record</b>—The following example shows that the new record (Key 2) column <b>From_Date</b> contains 2006.01.31 and the old record (Key 1) column <b>To_Date</b> contains this same value.</li></ul></li></ul> <table><thead><tr><th>Key</th><th>Empno</th><th>Name</th><th>Salary</th><th>From_Date</th><th>To_Date</th></tr></thead><tbody><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></tbody></table> <ul style="list-style-type: none"><li><b>Use one day before "valid from" date of new record</b>—The following example shows that the new record (Key 2) column <b>From_Date</b> contains 2006.01.31 and the old record (Key1) column <b>To_Date</b> contains a date that is one day before that value, 2006.01.30.</li></ul> <table><thead><tr><th>Key</th><th>Empno</th><th>Name</th><th>Salary</th><th>From_Date</th><th>To_Date</th></tr></thead><tbody><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><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></tbody></table>	Key	Empno	Name	Salary	From_Date	To_Date	1	100	Chang	10000.00	2006.01.01	2006.01.31	2	100	Chang	20000.00	2006.01.31	9000.12.31	Key	Empno	Name	Salary	From_Date	To_Date	1	100	Chang	10000.00	2006.01.01	2006.01.30	2	100	Chang	20000.00	2006.01.31	9000.12.31
Key	Empno	Name	Salary	From_Date	To_Date																																
1	100	Chang	10000.00	2006.01.01	2006.01.31																																
2	100	Chang	20000.00	2006.01.31	9000.12.31																																
Key	Empno	Name	Salary	From_Date	To_Date																																
1	100	Chang	10000.00	2006.01.01	2006.01.30																																
2	100	Chang	20000.00	2006.01.31	9000.12.31																																

Option	Description
<b>Preserve delete row(s) as update row(s)</b>	Converts DELETE rows to UPDATE rows in the target warehouse and, if you previously set effective date values ( <b>Valid from</b> and <b>Valid to</b> ), sets the <b>Valid To value</b> 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 <b>Detect deleted row(s) from comparison table</b> option selected.

### 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	Alvarez	81	UPDATE

Comparison table

EmpID	LastName	Rank	OpCode
1001	Alvarez	80	NORMAL
1009	Davis	80	NORMAL

History  
Preserving  
transform

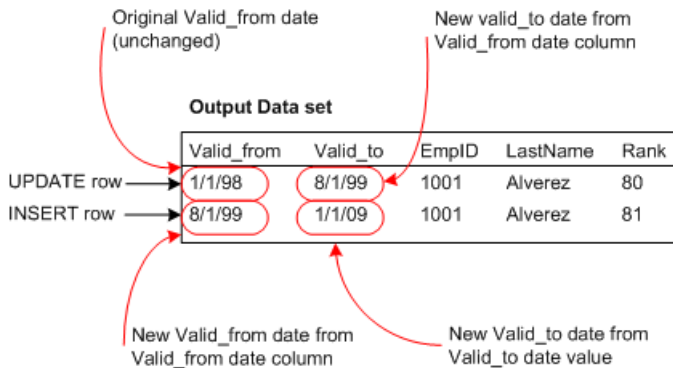


Output data set

EmpID	LastName	Rank	OpCode
1007	Chang	NULL	INSERT
1001	Alvarez	81	INSERT

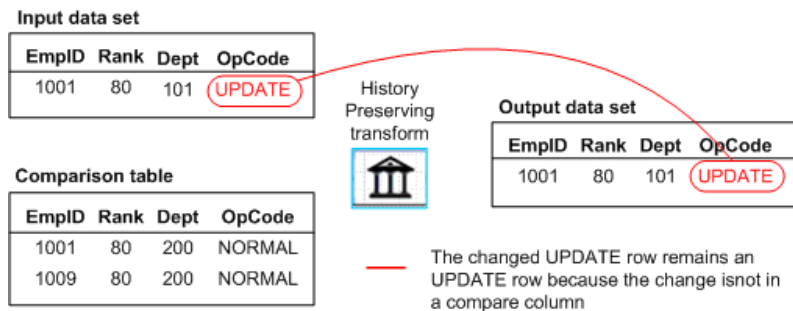
If you specified flag values for the History\_Preserving transform, Data Services includes the Flag set value in the INSERT row. In addition Data Services includes an UPDATE row to update the previously current row in the warehouse with the Flag reset value.

If you specified effective date columns (Valid to date column and Valid from date column), Data Services includes this data as well.



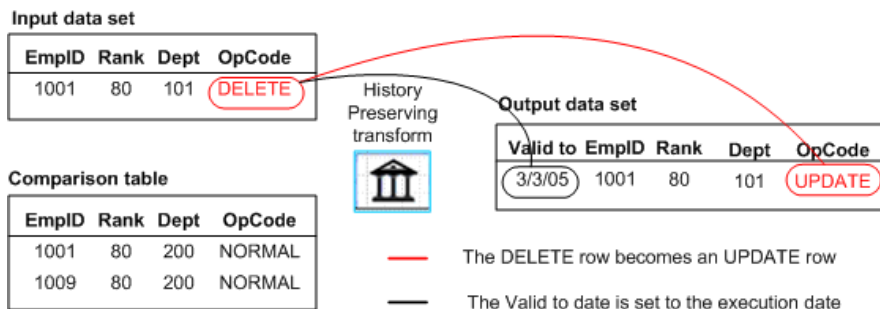
- An UPDATE row

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 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.





Nested schemas in the input are passed through without change.

## 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.

### 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.

### 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**.

### 5.3.6.3 Options

Option	Description																
<b>Generated key column</b>	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.																
<b>Increment value</b>	The interval between generated key values. For added flexibility, you can enter a variable for this option.																
<b>Table name</b>	<p>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. <b>Table name</b> 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:</p> <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>Sybase ASE</td><td>User name</td></tr> <tr> <td>Sybase IQ</td><td>User name</td></tr> </table>	Database type	Owner value	DB2	Data source dependent	Informix	Informix-defined user name	Microsoft SQL Server	User name	ODBC	Data source dependent	Oracle	User name	Sybase ASE	User name	Sybase IQ	User name
Database type	Owner value																
DB2	Data source dependent																
Informix	Informix-defined user name																
Microsoft SQL Server	User name																
ODBC	Data source dependent																
Oracle	User name																
Sybase ASE	User name																
Sybase IQ	User name																

### 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.

## 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 Topics

- [Designer Guide: Importing CDC data from Oracle](#)

### 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.

### 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.

### 5.3.7.3 Options

Option	Description
<b>Additional grouping columns</b>	In addition to the <b>Sequencing column</b> , 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.
<b>Define columns to filter updated rows</b>	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.
<b>Input already sorted by sequencing column</b>	<p>This transform by default assumes that the input data is already sorted based on the value selected in the <b>Sequencing column</b> box. If you deselect this check box, Data Services will re-sort the input data using the value in the <b>Sequencing column</b> box.</p> <p>Use the re-sort capability of this transform only when necessary as it impacts job performance.</p>
<b>Row operation column</b>	<p>(Required) Specifies a column with one of the following output operation codes for each row:</p> <ul style="list-style-type: none"> <li>• I for INSERT</li> <li>• B for before-image of an UPDATE</li> <li>• U for after-image of an UPDATE</li> <li>• D for DELETE</li> </ul> <p>If you are using a relational or mainframe CDC source table, the <code>DI_OPERATION_TYPE</code> column is automatically selected as the <b>Row operation column</b>.</p>

Option	Description
<b>Sequencing column</b>	<p>(Required) Specifies an integer used to order table rows.</p> <p>If you are using a relational or mainframe CDC source table, the <code>DI_SEQUENCE_NUMBER</code> column is automatically selected as the <b>Sequencing column</b>.</p>

### 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.

### 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.

### 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
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

Sequencing Column	Operation Column	Internal Operation Code
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

Sequencing Column	Operation Column	Internal Operation Code
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 Topics

- [Designer Guide: Using before-images](#)



### 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 selected 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.

## 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.

### 5.3.8.1 Data inputs

A data set with rows flagged as NORMAL.

### 5.3.8.2 Options

Option	Description
<b>Data field column</b>	<p>The name of the column that contains the pivoted data. This column contains all of the <b>Pivot columns</b> values.</p> <p>The data type of this column is determined by the data type of <b>Pivot columns</b>. If two or more <b>Pivot columns</b> 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.</p>
<b>Header column</b>	The name of the column that contains the pivoted column names. This column lists the names of the columns where the corresponding data originated.
<b>Non-pivot columns</b>	The columns in the source that are to appear in the target without modification.
<b>Pivot columns</b>	A set of columns to be rotated into rows. Describe these columns in the <b>Header column</b> . Describe the data in these columns in the <b>Data field column</b> .
<b>Pivot sequence column</b>	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.
<b>Pivot set</b>	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 <b>Data field column</b> and the <b>Header column</b> . Data Services automatically saves this information.

### 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.

### 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
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

Emp_name	Seq	Expense_Type	Expenses	Day_Type	Num_Days
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

### 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.

#### 5.3.9.1 Data inputs

A data set with rows flagged as NORMAL.

### 5.3.9.2 Options

Option	Description
<b>Axis value</b>	The value of the pivot axis column that represents a particular set of output columns. A set of <b>Pivoted columns</b> is generated for each axis value. There should be one <b>Axis value</b> for each unique value in the <b>Pivot axis column</b> .
<b>Column Prefix</b>	Text added to the front of the <b>Pivoted column</b> names when creating new column names for the rotated data. An underscore separates the prefix name from the pivoted column name.
<b>Default value</b>	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.
<b>Duplicate value</b>	Action taken when a collision occurs. A collision occurs when there is more than one row with the same key and value in the <b>Pivot axis column</b> . In this case, you can store either the first row or the last row, or you can abort the transform process.
<b>Input data is grouped</b>	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.
<b>Non-pivot columns</b>	The columns in the source table that will appear in the target table without modification.
<b>Pivot axis column</b>	The column that determines what new columns are needed in the output table. At run time, a new column is created for each <b>Pivoted column</b> and each unique value in this column.
<b>Pivoted columns</b>	The columns containing data you want rotated into the same row. A set of columns will be created for each unique value in the <b>Pivot axis column</b> .

### 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.



### 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

### 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 Topics

- [Designer Guide: Distributed data flow execution](#)
- [Teradata](#)

### 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 Topics

- [Operation codes](#)

### 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
<b>Compare columns</b>	<p>(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.</p> <p>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.</p> <p>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.</p>

Option	Description
Comparison method	<p>Select a method for accessing the comparison table:</p> <ul style="list-style-type: none"> <li>• <b>Row-by-row select</b> — 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.</li> <li>• <b>Cached comparison table</b> — 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 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.</li> <li>• <b>Sorted input</b> — Select this option to read the comparison table in the order of the primary key column(s) using sequential read.</li> </ul> <p>This option improves performance because Data Services reads the comparison table only once.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>

Option	Description
<p><b>Detect deleted row(s) from comparison table</b></p>	<p>(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.</p> <p><b>Note:</b> The Table_Comparison transform can flag rows as DELETE. However, rows actually become DELETE rows only after the data flow completes all other processing</p> <ul style="list-style-type: none"> <li>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: <ul style="list-style-type: none"> <li><b>Generated key column</b></li> <li><b>Detect deleted row(s) from comparison table</b></li> <li><b>Row-by-row select</b> or the <b>Sorted input</b> comparison method</li> </ul> </li> </ul> <p>If you choose the above options, choose one of the following for deleted rows with the same key value:</p> <ul style="list-style-type: none"> <li><b>Detect all rows:</b> Output detected DELETES for all rows.</li> <li><b>Detect row with largest generated key value:</b> Output detected DELETES for only the generated key row with the largest value.</li> </ul> <ul style="list-style-type: none"> <li>If you select this option and the <b>Cached comparison table</b> option, Data Services always deletes the row with largest generated key value.</li> </ul> <p>If you do not specify a <b>Generated key column</b>, then the sub-options with the <b>Detect Deleted row(s) from comparison table</b> option are not enabled. If Data Services finds duplicate keys in the comparison table, then the DELETE output is corrupted.</p> <p><b>Note:</b> If you choose the <b>Detect Deleted row(s) from comparison table</b> option, then the performance of the data flow will be slower. The comparison method most affected is the <b>Row-by-row select</b>, followed by <b>Cached compare</b> table, then <b>Sorted input</b> option. For <b>Row-by-row select</b> and <b>Cached compare</b> table, Data Services processes the deleted rows at the end of the data flow. For <b>Sorted input</b>, Data Services processes deleted rows as they are encountered in the data flow.</p> <p><b>Run as a separate process</b></p> <p>Creates a separate sub data flow process for the Table_Comparison transform.</p>

Option	Description
<b>Generated Key Column</b>	<p>(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.</p> <p>The Generated key column option insures that:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• If the input data set and the comparison table both have the column you specified in <b>Generated key column</b>, the transform does not compare the values for this column; it preserves the value.</li> </ul> <p>Without this option:</p> <ul style="list-style-type: none"> <li>• If your comparison table contains rows with the same primary keys, the transform arbitrarily chooses which of these rows to compare.</li> </ul>
<b>Input contains duplicate keys</b>	<p>(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 <b>Input Primary Key</b> 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.</p> <p>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.</p> <p>This option uses additional memory to track rows with duplicate keys. It is recommended that you use this feature only as needed.</p>

Option	Description														
<b>Input primary key column(s)</b>	<p>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.</p> <p>Drag the column(s) from the input schema into the <b>Input primary key columns</b> box. The transform selects rows from the comparison table that match the values from the primary key columns in the input data set.</p> <p>If values from more than one column are required to uniquely specify each row in the table, add more than one column to the <b>Input primary key columns</b> box.</p> <p>You cannot include nested schemas in the <b>Input primary key columns</b> list.</p>														
<b>Table name</b>	<p>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. <b>Table name</b> 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:</p> <table data-bbox="651 1045 1439 1436"> <tr> <th data-bbox="651 1045 1043 1098">Database type</th><th data-bbox="1043 1045 1439 1098">Owner value</th></tr> <tr> <td data-bbox="651 1098 1043 1155">DB2</td><td data-bbox="1043 1098 1439 1155">Data source dependent</td></tr> <tr> <td data-bbox="651 1155 1043 1211">Informix</td><td data-bbox="1043 1155 1439 1211">Informix-defined user name</td></tr> <tr> <td data-bbox="651 1211 1043 1268">Microsoft SQL Server</td><td data-bbox="1043 1211 1439 1268">User name</td></tr> <tr> <td data-bbox="651 1268 1043 1325">ODBC</td><td data-bbox="1043 1268 1439 1325">Data source dependent</td></tr> <tr> <td data-bbox="651 1325 1043 1381">Oracle</td><td data-bbox="1043 1325 1439 1381">User name</td></tr> <tr> <td data-bbox="651 1381 1043 1436">Sybase</td><td data-bbox="1043 1381 1439 1436">User name</td></tr> </table>	Database type	Owner value	DB2	Data source dependent	Informix	Informix-defined user name	Microsoft SQL Server	User name	ODBC	Data source dependent	Oracle	User name	Sybase	User name
Database type	Owner value														
DB2	Data source dependent														
Informix	Informix-defined user name														
Microsoft SQL Server	User name														
ODBC	Data source dependent														
Oracle	User name														
Sybase	User name														



Option	Description
<b>Filter</b>	<p>(Optional) Limits the rows from the comparison table that are considered for comparison against the input data set.</p> <p><b>Caution:</b> Incorrect filtering can produce false <code>INSERT</code> rows. Construct your filter expression carefully to avoid unexpected results.</p> <p>Valid filter expressions may contain the following elements:</p> <ul style="list-style-type: none"> <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>&lt;</code>, <code>&gt;</code>, <code>&lt;=</code>, <code>&gt;=</code>, <code>LIKE</code>, <code>IN</code>)</li> <li>• Logical operators (<code>AND</code>, <code>OR</code>)</li> </ul> <p><b>Note:</b> Data Services functions, and <code>JOIN</code> and other SQL statements can not be used in filter expressions.</p> <p>Examples of valid filter expressions:</p> <ul style="list-style-type: none"> <li>• <code>TC.col1 IN (1, 2, 3)</code></li> <li>• <code>TC.col1 &gt;= 1 AND TC.col1 &lt; 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>

### 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

EmplID	LastName	OpCode
1001	Alvarez	NORMAL
1007	Chang	NORMAL
1009	Davis	NORMAL



Output data set

EmplID	LastName	Rank	OpCode
1007	Chang	NULL	INSERT

Comparison table

EmplID	LastName	Rank	OpCode
1001	Alvarez	80	NORMAL
1009	Davis	80	NORMAL

- 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.

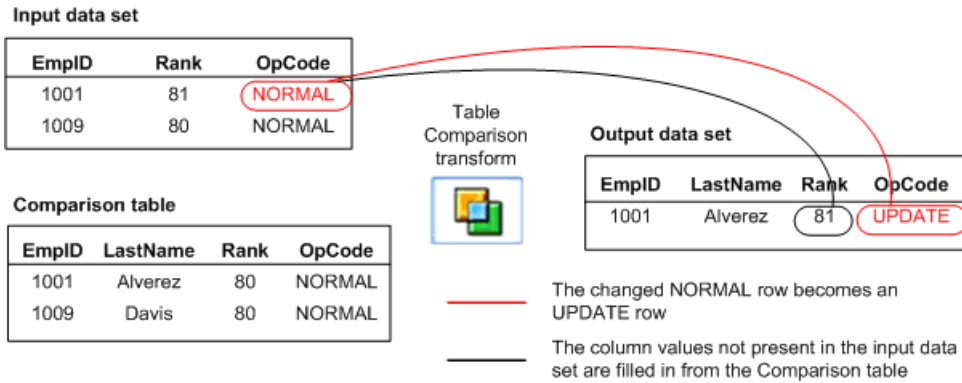
To avoid inserting rows when the data differs only by number of trailing blanks, take either of the following actions:

- 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.



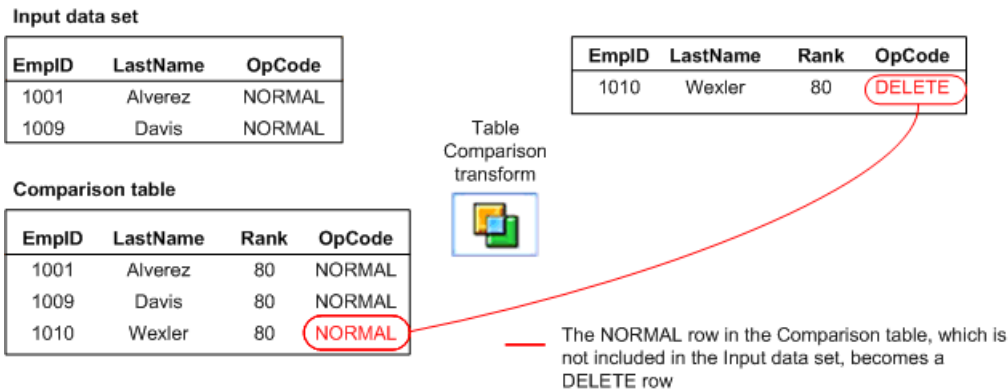
### Note:

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.



- 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 Topics

- [Operation codes](#)

## 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.

### 5.3.11.1 Data inputs

XML file or XML message.

You can connect more than one XML\_Pipeline transform to an XML source.



### 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

### 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.

### 5.3.11.4 Limitations

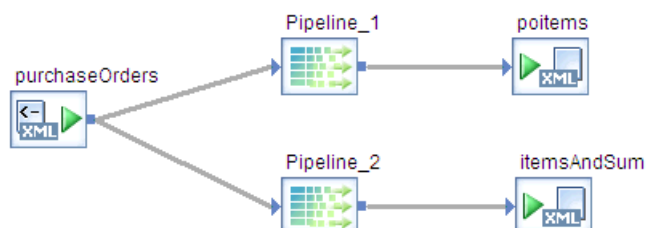
## Limitations

The XML\_Pipeline transform:

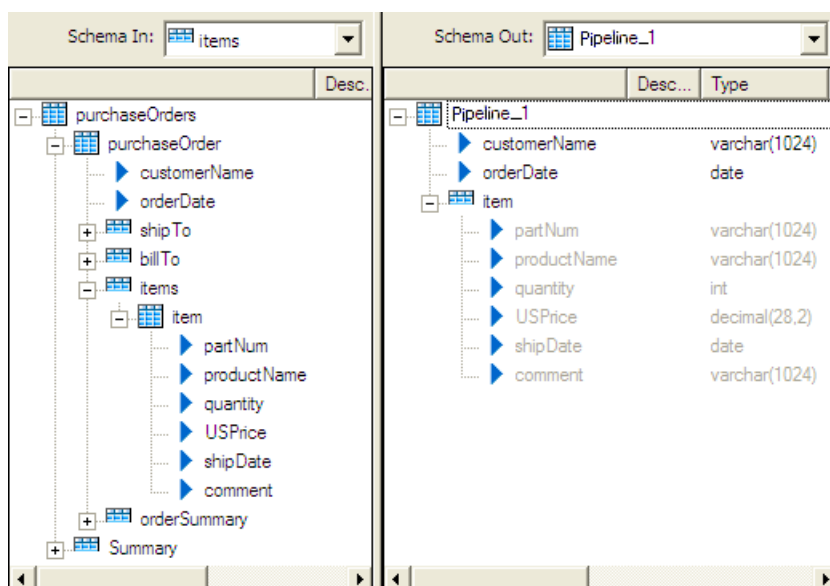
- Pushes operations to the XML source during transform execution, so you cannot use a breakpoint between your XML source and a XML\_Pipeline transform.

### 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.

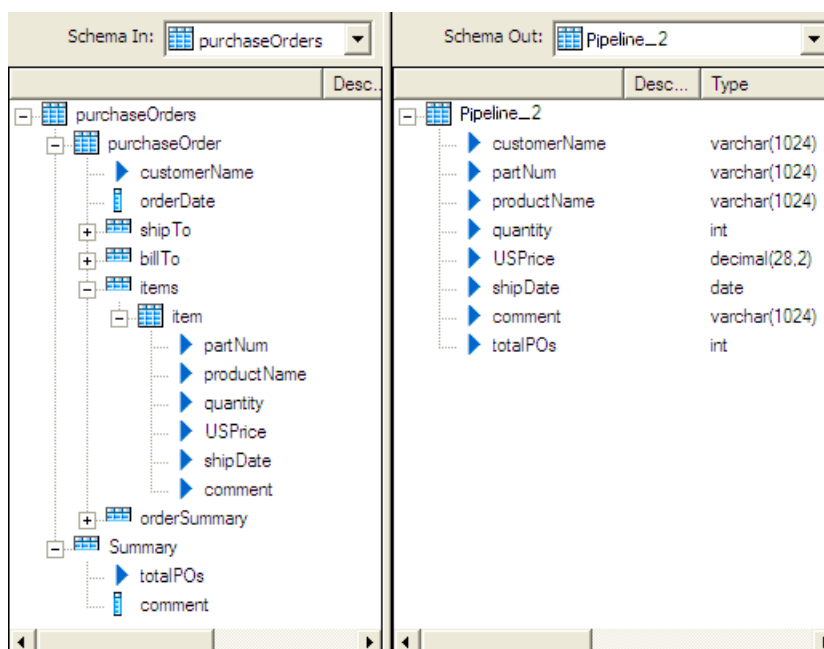


- The "XML\_Pipeline\_1" transform allows XML data to flow because the repeatable column, purchaseOrders.purchaseOrder.items.item is selected. The XML source produces one row after parsing one item.



- 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.



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 Topics

- [Query](#)

## 5.4 Data Quality transforms

### 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 5-54: Associate transform

Transform configuration	Description
AssociateGroupStatistics_AssociateBatch	A sample Associate transform configured with group statistics.
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 5-55: Country ID transform

Transform configuration	Description
CountryID2Char	A sample Country ID transform configured to generate the two-character ISO country code.

Table 5-56: 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.



Transform configuration	Description
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 5-57: 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 5-58: 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.

Transform configuration	Description
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.
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 5-59: 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 5-60: 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.
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.

Transform configuration	Description
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 5-61: 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.
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.

Transform configuration	Description
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 5-62: User-Defined transform

Transform configuration	Description
Base_UserDefined	The base User-Defined configuration that is used to configure a User-Defined transform.

**Related Topics**

- [Designer Guide: Data Flows, To create a transform configuration](#)

## 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 BusinessObjects 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 BusinessObjects 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 BusinessObjects 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 BusinessObjects Data Services to run the blueprints.

### 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	<p>Option_Content_Domain_Sequence</p> <p>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.</p> <p>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.</p> <p>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.</p>
Data Cleanse	<p>Option_Output_Format</p> <p>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.</p>

Transform	Dynamic input field
Global Address Cleanse	Option_Canada_Output_Address_Language
Global Address Cleanse	Option_GAC_Dual_Address
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_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

## 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.

### 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 dataflows where the schema is used. You cannot change the content type in an XML Schema from within a single dataflow.

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
- Prenom
- Prenom Match Std

- Region
- Sortcode Rte
- SSN
- Title
- Title Match Std

## 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 Topics

- [Associate transform options](#)
- [Association options](#)
- [Designer Guide: Match, Association matching](#)

## 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 Topics

- [Transform configurations](#)
- [Downloading blueprints and other content objects](#)

## 5.4.5.2 Associate transform options

The following options control transform-level functions.

Option	Description
<b>Associate set name</b>	Specifies the name for your Associate set.
<b>Generate report data</b>	<p>Specifies whether to generate report data for this transform. This option is available in every transform that generates report data.</p> <p><b>Yes:</b> Generates report data for this transform.</p> <p><b>No:</b> 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.</p>
<b>Logical source field</b>	Specifies the field that contains the ID for the logical source.
<b>Physical source field</b>	Specifies the field that contains the ID for the physical source (Reader).
<b>Run as a separate process</b>	<p>Specifies whether to split transform into a sub-data flow that can improve performance and throughput by using separate memory and computer resources.</p> <p><b>Yes:</b> Splits transform into sub-data flow that runs on a separate process.</p> <p><b>No:</b> Keeps transform in same process as the rest of the dataflow.</p>

**Related Topics**

- [Performance Optimization Guide: Distributing Data Flow Execution, Run as a separate process](#)

**5.4.5.3 Association options**

This is a repeating option group: you can add as many associations as you want.

Option	Description
<b>Association name</b>	Specifies the name for this association. Be sure that it is unique among other associations in this transform.
<b>Group Number field</b>	Specifies the field that contains the group number data from the Match transform. Add as many of these as you want by clicking the <b>Add Row</b> button.

**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 Topics

- [Best record options: Best Record tab](#)
- [Group statistics options](#)
- [Group prioritization options: Priority Order tab](#)
- [Unique ID options: Unique ID tab](#)
- [Output flag selection options](#)
- [Designer Guide: Match, Best record](#)
- [Designer Guide: Match, Unique ID](#)

### *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
<b>Best record name</b>	Enter a name for your Best Record operation. Make sure that this name is unique within this Match transform.

Option	Description
<b>Best record strategy</b>	<p>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).</p> <p><b>Custom:</b> 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.</p> <p><b>Date:</b> Choose Date to base your strategy on a date field.</p> <p><b>Length:</b> Choose Length to base your strategy on the length of data in a field.</p> <p><b>Non_Blank:</b> Choose Non_Blank to base your strategy on the completeness of data in a field.</p> <p><b>Priority_Number:</b> Choose Priority_Number to base your strategy on a number.</p> <p><b>Priority_String:</b> Choose Priority_String to base your strategy on string data in a field.</p>
<b>Strategy priority</b>	<p>These are the choices for priorities for each of the best record strategies, other than Non_Blank and Custom.</p> <p>Date</p> <ul style="list-style-type: none"> <li>• <b>Newest:</b> The newest date in the field will cause an action to take place.</li> <li>• <b>Oldest:</b> The oldest date in a field will cause an action to take place.</li> </ul> <p>Length</p> <ul style="list-style-type: none"> <li>• <b>Longest:</b> The longest string in a field will cause an action to take place.</li> <li>• <b>Shortest:</b> The shortest string in a field will cause an action to take place.</li> </ul> <p>Priority_Number</p> <ul style="list-style-type: none"> <li>• <b>Highest:</b> The highest number in a field will cause an action to take place.</li> <li>• <b>Lowest:</b> The lowest number in a field will cause an action to take place.</li> </ul> <p>Priority_String</p> <ul style="list-style-type: none"> <li>• <b>Ascending:</b> The string with the most ascending string order will cause an action to take place.</li> <li>• <b>Descending:</b> The string with the most descending string order will cause an action to take place.</li> </ul>
<b>Strategy field</b>	Choose a field that contains data that you need to execute your strategy.

Option	Description
<b>Posting destination</b>	<p>Specifies the destination record.</p> <p><b>Master:</b> Post only to a master record.</p> <p><b>Subs:</b> Post only to subordinate records.</p> <p><b>Master to Subs:</b> Push information from the master record and post it to each subordinate record.</p> <p><b>All:</b> Post to both the master and subordinate records.</p>
<b>Post only once per destination</b>	<p><b>Yes:</b> Post only once per destination record. After data is posted to the destination record, the operation stops.</p> <p><b>No:</b> 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.</p> <p>Set this option to Yes when you are using the NON_BLANK strategy.</p> <p>Set this option to No when you are using the Longest, Shortest, Newest, Oldest, Ascending, or Descending priorities.</p> <p><b>Note:</b> This option is ignored when using the <b>Master to Subs</b> posting destination. With this posting destination, information can be posted only once.</p>
<b>View/Edit Python</b>	<p>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.</p>

### 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
<b>Source field</b>	Specifies the name of the source field in the input record.
<b>Destination field</b>	<p>Specifies the name of the destination field in the output record, or the destination of this best record action.</p> <p>You can have the action post to the same input field, or you can post to a different field.</p>

Option/Option group	Description
<b>Custom</b>	<p><b>Yes:</b> Specifies that you want to create custom Python code to perform an action on the destination field.</p> <p><b>No:</b> Specifies that you want to use the same source and destination fields.</p> <p>When this option is set to No, the contents of the source field are copied to the destination field.</p>
<b>Editor</b>	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 Topics

- [Designer Guide: Match, Best record](#)

### *Best record options: Destination Protection tab*

Protect data from being changed by enabling and defining destination protection.

Option	Description
<b>Best record name</b>	Enter a name for your Best Record operation. Make sure that this name is unique within this Match transform.
<b>Enable destination protection</b>	Select to protect records from best record operations that may modify the contents.
<b>Default destination protection</b>	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.
<b>Specify destination protection by field</b>	Select to enable destination protection through a value in a field.
<b>Destination protection field</b>	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.
<b>Specify destination protection by source</b>	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
<b>Source Name</b>	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.
<b>Destination protected</b>	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.

### *Group statistics options*

The Group Statistics option group includes the following options:

Option	Description
<b>Group statistics name</b>	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.
<b>Generate only basic statistics</b>	Select if you want to generate match group statistics. These will not include any statistics about input sources.
<b>Generate source statistics from input sources</b>	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.
<b>Generate source statistics from source values</b>	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.
<b>Logical source field</b>	Specifies the field that holds the value for your logical sources.
<b>Default logical source value</b>	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.
<b>Count all sources</b>	Select to count all sources, no matter what the value in the Logical source field is.
<b>Choose sources to count</b>	Select to specify particular sources to count, based on values in the Logical source field.

Option	Description
<b>Default count flag value</b>	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. <b>Yes:</b> Counts all of the records in the source. <b>No:</b> Does not count any of the records in the source.
<b>Auto generate sources</b>	Select to generate sources based on the value in a field.
<b>Predefined count flag field</b>	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
<b>Source value</b>	Specifies the value in the field to find. This value is case sensitive.
<b>Count</b>	Specifies whether you want to use the value you entered in the Logical source value option in the count. <b>Yes:</b> Includes the logical source value in the count. <b>No:</b> Does not include the value in the logical source value option in the count.

### 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
<b>Prioritization name</b>	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
<b>Input field</b>	Choose a field to sort your records on.
<b>Field order</b>	Specifies in which order records should be sorted.

### Group prioritization options: Record Completeness tab

Option	Description
<b>Prioritization name</b>	Specifies the name for this Group prioritization operation. If you have multiple operations in this Match transform, be sure to make this name unique.
<b>Order records based on completeness of data</b>	Select this option to apply priority and blank penalty points to records to help control the order of your records.
<b>Define only field penalties</b>	Select this option when penalties need to be assessed based on blank fields.
<b>Define priority and penalty fields</b>	Select this option when you have specific fields that contain the actual integer values for priority and blank penalty.
<b>Record priority field</b>	Choose the field that contains priority values. This field must contain an integer.
<b>Apply blank penalty field</b>	Choose the field that contains the indicator (Y or N) for applying blank penalty points to a record.
<b>Define priority and penalty based on input source</b>	Select to have your record priority and blank penalty indicator (Y or N) determined by membership in a given source.
<b>Source Name</b>	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.
<b>Priority</b>	Type a priority value (an integer) in the Priority column. Remember that the lower the priority score, the higher the priority.
<b>Apply Blank Penalty</b>	Choose Yes or No to determine whether a blank penalty is applied to a record based on membership to this source.

Option	Description
<b>Default record priority</b>	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.
<b>Default apply blank penalty</b>	<p>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 if a record does not belong to any of the sources specified.</p> <p><b>Yes:</b> 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.</p> <p><b>No:</b> No penalty is applied when the fields are blank.</p>
<b>Input field</b>	Displays the input fields available to assign a blank penalty score to.
<b>Blank penalty</b>	Assign a penalty value (an integer) to apply when the specified field is blank in a record.

### *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
<b>Unique ID name</b>	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.

Option	Description
<b>Processing operation</b>	<p>Specifies the type of processing operation you want the application to perform. Valid values include:</p> <p><b>Assign:</b> 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).</p> <p><b>AssignCombine:</b> 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).</p> <p><b>Combine:</b> 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).</p> <p><b>Delete:</b> Removes unique IDs from records that have one, unless they are protected.</p> <p><b>Split:</b> 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).</p>
<b>Recycle unique IDs</b>	<p>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:</p> <p><b>Yes:</b> Recycle freed-up unique IDs.</p> <p><b>No:</b> Do not recycle freed-up unique IDs.</p>
<b>ID field</b>	<p>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.</p>
<b>Field</b>	<p>The starting unique ID is obtained from an input field.</p> <p>Be sure to map in a field from an upstream transform before you add this option.</p>
<b>Starting unique ID field</b>	<p>Choose the field that passes in the starting unique ID. If no Unique ID is received, the starting number will default to 1.</p>
<b>Constant value</b>	<p>The starting ID is specified as a positive whole number in the Starting value option.</p>

Option	Description
<b>Starting value</b>	Indicates the starting unique ID value. Valid values range from 1 to UINT_MAX (unsigned integer max). The default value is 1.
<b>Value from file</b>	The starting Unique ID is read from the file specified in the File option.
<b>File</b>	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.
<b>GUID</b>	<p>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: <code>TimeLow-TimeMid-TimeHighAndVersion-ClockSeqAndReservedClockSeqLow-Node</code>.</p> <p>For more information about UUID, see the RFC document in the Related Topics section.</p>
<b>Save ending ID to file and reclaim recycled IDs</b>	<p>Specifies whether to save the last ID that was assigned to a file.</p> <p>Additionally, specifies whether to reclaim recycled IDs.</p>
<b>File</b>	Specifies the file to write the last assigned ID to.
<b>Allow multiple Match transforms to access unique ID file</b>	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 dataflow to share a single unique ID file.
<b>Number of IDs to get when accessing file</b>	<p>Specifies the number of IDs to retrieve from the unique ID file during each access.</p> <p>For example, with a setting of 100, the first process will access the file and retrieve IDs numbered 1-100. The next process will 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.</p> <p><b>Note:</b></p> <p>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.</p>

Option	Description
<b>Group number field</b>	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 Topics

- [Designer Guide: Match, Assign unique IDs using a file](#)
- [Designer Guide: Match, Unique ID](#)
- [UUID RFC: http://www.ietf.org/rfc/rfc4122.txt](http://www.ietf.org/rfc/rfc4122.txt)

### 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
<b>Unique ID name</b>	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.
<b>Enable destination protection</b>	Select if you want to protect a destination source from having its unique IDs overwritten with the IDs from matching records.
<b>Default destination protection</b>	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.
<b>Specify destination protection by field</b>	Select to enable destination protection through a value in a field.
<b>Unique ID protected field</b>	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.
<b>Specify destination protection by source</b>	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
<b>Source name</b>	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.
<b>Unique ID protected</b>	<b>Yes:</b> This source is protected. <b>No:</b> This source is not protected.

#### 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.

#### 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 dataflow, 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 dataflow 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 Topics

- [Input Fields](#)
- [Output fields](#)
- [Transform configurations](#)

## 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.

### 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. <b>1010</b> : Indicates a tie in identifying the country. <b>1005</b> : 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 Topics

- [Input Fields](#)

### 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.

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.

### 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 Topics

- [Transform configurations](#)
- [Downloading blueprints and other content objects](#)

### 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.

#### 5.4.7.2.1 Common options

Option	Description
Run As Separate Process	<b>Yes:</b> Splits the transform into a separate process. <b>No:</b> Keeps the transform in the same process as the rest of the data flow.

#### 5.4.7.2.2 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	

Option	Description
	<p>A content domain specifies which domain's properties should be assigned to a variation. You can specify more than one content domain.</p> <p>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.</p> <p>Select the content domains you want to include. The arrows allow you to change the order of the content domains.</p> <p>GLOBAL - Global</p> <p>AR - Arabic</p> <p>ZH - Chinese</p> <p>CS - Czech</p> <p>DA - Danish</p> <p>NL - Dutch</p> <p>EN_US - English (United States &amp; Canada)</p> <p>EN_GB - English (United Kingdom &amp; Ireland)</p> <p>EN_AU - English (Australia &amp; New Zealand)</p> <p>EN_IN - English (India)</p> <p>FR - French</p> <p>DE - German</p> <p>HU - Hungarian</p> <p>ID - Indonesian</p> <p>IT - Italian</p> <p>JA - Japanese</p> <p>MS - Malay</p> <p>NO - Norwegian</p> <p>PL - Polish</p> <p>PT_BR - Portuguese (Brazil)</p> <p>PT_PT - Portuguese (Portugal)</p>

Option	Description
	<p>RO - Romanian</p> <p>RU - Russian</p> <p>SK - Slovak</p> <p>ES_MX -Spanish (Latin America)</p> <p>ES_ES - Spanish (Spain)</p> <p>SV - Swedish</p> <p><b>Note:</b> You can set this option as a dynamic input field.</p>

Option	Description
Output Format	



Option	Description
	<p>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.</p> <p>Valid values for this option are:</p> <p><b>AR</b> Arabic</p> <p><b>ZH</b> Chinese</p> <p><b>CS</b> Czech</p> <p><b>DA</b> Danish</p> <p><b>NL</b> Dutch</p> <p><b>EN_US</b> English (United States &amp; Canada)</p> <p><b>EN_GB</b> English (United Kingdom &amp; Ireland)</p> <p><b>EN_AU</b> English (Australia &amp; New Zealand)</p> <p><b>EN_IN</b> English (India)</p> <p><b>FR</b> French</p> <p><b>DE</b> German</p> <p><b>HU</b> Hungarian</p> <p><b>ID</b> Indonesian</p> <p><b>IT</b> Italian</p> <p><b>JA</b> Japanese</p> <p><b>MS</b> Malay</p> <p><b>NO</b> Norwegian</p> <p><b>PL</b> Polish</p> <p><b>PT_BR</b> Portuguese (Brazil)</p> <p><b>PT_PT</b> Portuguese (Portugal)</p> <p><b>RO</b> Romanian</p> <p><b>RU</b> Russian</p> <p><b>SK</b> Slovak</p> <p><b>ES_MX</b> Spanish (Latin America)</p>

Option	Description
	<b>ES_ES</b> Spanish (Spain) <b>SV</b> Swedish <b>Note:</b> You can set this option as a dynamic input field.

### Related Topics

- [Dynamic transform settings](#)
- [Data Services Designer Guide: Ordered options editor](#)
- [Information Steward User's Guide: About domains](#)
- [Information Steward User's Guide: About output format](#)

#### 5.4.7.2.3 Options

The Options group includes settings that control how the Data Cleanse transform parses and outputs data.

Option	Description
Filter Output Fields	<p>Specifies which output fields are displayed in the Output tab.</p> <p><b>Show_Relevant_Fields:</b> 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 <i>Extra</i> fields are always available.</p> <p><b>Show_All_Fields:</b> All Data Cleanse transform fields are available.</p>

#### 5.4.7.2.4 Input word breaker

Controls how the parser breaks input data.

Option	Description
Break on Whitespace Only	<p>Specifies whether the Data Cleanse transform breaks input data only on white space or on white space, punctuation, alphanumeric transitions, and script transitions.</p> <p><b>Yes:</b> Input data breaks only on white space.</p> <p><b>No:</b> 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.</p> <p>This option allows the Data Cleanse transform to recognize alphanumeric product codes as entries in a custom cleansing package. For example, if <b>Break on Whitespace only</b> is set to no, the parser breaks a product code such as AF302 into two tokens, AF and 302. If <b>Break on Whitespace only</b> is set to yes, the parser recognizes AF302 as a single entry.</p> <p><b>Note:</b></p> <p>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.</p>

#### 5.4.7.2.5 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	<p>Specifies whether the transform should include assigned prenames (for example, Mr. or Mrs.) in the Prenom output field.</p> <p>The Prenom 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 <b>Gender Options &gt; Use Given Name2 To Assign Gender</b> and <b>Gender Options &gt; Use Family Name to Assign Gender</b> options.</p> <p><b>Yes:</b> Turns on prename assignment.</p> <p><b>No:</b> Turns off prename assignment. The Prenom output field contains only prenames included in the input data</p>
Associate Name Title	<p>Defines how name and occupational title data found in separate input fields are associated.</p> <p><b>Yes:</b> Data Cleanse assumes that the name and title data describe the same person and is associated.</p> <p><b>No:</b> Data Cleanse assumes that the name and title data is not associated.</p> <p>For example, the diagram below shows the difference in the output based on the <b>Associate Name Title</b> setting.</p> <div data-bbox="518 1144 1236 1470"> <pre> graph LR     Input["Name_Line1: Thomas Jones Name_Line2: Manager"] --&gt; Setting     subgraph Setting_Box [Setting]         Yes[Yes]         No[No]     end     Setting_Box -- Yes --&gt; Output_Yes     Setting_Box -- No --&gt; Output_No     subgraph Output_Yes [Output]         P1P[Person1_Person: Thomas Jones]         P1T[Person1_Title: Manager]         P2P[Person2_Person: &lt;Null&gt;]         P2T[Person2_Title: &lt;Null&gt;]     end     subgraph Output_No [Output]         P1P2[Person1_Person: Thomas Jones]         P1T2[Person1_Title: &lt;Null&gt;]         P2P2[Person2_Person: &lt;Null&gt;]         P2T2[Person2_Title: Manager]     end </pre> </div>

Option	Description
Combine Compound Names	<p>Specifies how compound family names are standardized when the family name includes a Pre_Family_Name that is also a Pre_Family_Name_Combine.</p> <p><b>Yes:</b> Combines compound family names. For example, the family name Mc Donald would combine to McDonald.</p> <p><b>No:</b> Retains a space between the Pre_Family_Name and the family name. For example, the family name Mc Donald remains Mc Donald.</p> <p><b>Note:</b> 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.</p>
Enable Presumptive Name Parsing	<p>Specifies whether you want to use presumptive name parsing on Name_Line input fields.</p> <p><b>Yes:</b> Turns on presumptive name parsing. Data in the Name_Line input field is treated as a name.</p> <p><b>No:</b> 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.</p> <p>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.</p>

Option	Description
Name Order	<p>Defines how Data Cleanse applies parsing rules to determine the content of the Given_Name and Family_Name output fields.</p> <p><b>Given_Family_Name_Strict</b> and <b>Family_Given_Name_Strict</b>: 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><b>Given_Family_Name_Suggest</b> and <b>Family_Given_Name_Suggest</b>: 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><b>Unknown</b>: 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>

#### 5.4.7.2.6 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**.

**Note:**

The options in this group apply only to person and firm cleansing packages.

Option	Description
Use Given Name2 To Assign Gender (Gender Options)	<p>When the gender of the prename and Given_Name1 are unassigned or ambiguous, assigns gender based on the gender of the parsed Given_Name2.</p> <p><b>Yes</b>: Turns on the option.</p> <p><b>No</b>: Turns off the option.</p> <p>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.</p>

Option	Description
Use Family Name To Assign Gender (Gender Options)	<p>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.</p> <p><b>Yes:</b> Turns on the option.</p> <p><b>No:</b> Turns off the option.</p> <p>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.</p>

#### 5.4.7.2.7 Firm standardization options

The firm standardization options control how the Data Cleanse transform standardizes firm-related output.

**Note:**

The options in this group apply only to person and firm cleansing packages.

Option	Description
Enable Presumptive Firm Parsing	<p>Specifies whether you want to use presumptive firm parsing on Firm_Line input fields.</p> <p><b>Yes:</b> Turns on presumptive firm parsing. Data in the Firm_Line input field is treated as a firm name.</p> <p><b>No:</b> 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.</p> <p>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.</p>

#### 5.4.7.2.8 Other standardization options

Standardization options control how the Data Cleanse transform standardizes many types of output.

Option	Description
Capitalization	<p>Specifies the casing of your output.</p> <p><b>Lower:</b> Converts the output to lowercase. For example, john mckay.</p> <p><b>Mixed:</b> 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.</p> <p>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.</p> <p><b>Preserve:</b> Preserves the input casing.</p> <p><b>Upper:</b> Converts the output to uppercase. For example, JOHN MCKAY.</p>
Character Width Style	<p>Specifies the character width used in output fields. Useful when processing Japanese or mixed language data.</p> <p><b>Normal_Width:</b> 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.</p> <p><b>Full-width:</b> 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.</p> <p><b>Half-width:</b> 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.</p> <p><b>Note:</b></p> <p>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.</p> <p>For template tables, selecting the <b>Use NVARCHAR for VARCHAR columns in supported databases</b> box changes the VARCHAR column type to NVARCHAR and allows for increased data size.</p>



Option	Description
North American Phone Delimiter	<p>Specifies a character to use as a delimiter for phone output following the North American Numbering Plan (NANP).</p> <p><b>Backslash (\):</b> Uses backward slashes as the delimiter in the phone number. For example, 123\656\5000.</p> <p><b>Dash (-):</b> Uses dashes as the delimiter in the phone number. For example, 123-656-5000.</p> <p><b>Slash (/):</b> Uses forward slashes as the delimiter in the phone number. For example, 123/656/5000.</p> <p><b>None:</b> Does not add a delimiter to the phone number. For example, 1236565000.</p> <p><b>Period (.):</b> Uses periods as the delimiter in the phone number. For example, 123.656.5000.</p> <p><b>Space:</b> Uses spaces as the delimiter in the phone number. For example, 123 656 5000.</p>
North American Phone Delimiter After Area	<p>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.</p> <p><b>Yes:</b> Adds a delimiter. For example, 123-656-5000.</p> <p><b>No:</b> Does not add a delimiter. For example, 123 656-5000.</p>
North American Phone Parens Area	<p>Controls placement of parentheses ( ) around the area code of phone number output following the North American Numbering Plan (NANP).</p> <p><b>Yes:</b> Includes the parentheses. For example, (123) 656-5000.</p> <p><b>No:</b> Omits the parentheses. For example, 123 656-5000.</p>

Option	Description
One-to-one mapping	<p>Specifies whether to place the input data into the corresponding output field for the following parsers: Phone, Email, Date.</p> <p><b>Yes:</b> 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.</p> <p><b>No:</b> 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.</p>
Phone Extension Text	Specifies the standard text for a phone extension. For example, Ext.
Remove Punctuation	<p>Removes all punctuation from standardized data (with the exception of hyphens between names).</p> <p><b>Yes:</b> Removes punctuation.</p> <p><b>No:</b> Leaves the punctuation as is on input.</p> <p>For example, if the standard form for extra large is X.L. and the option is set to Yes, the standardized output becomes XL.</p>
SSN Delimiter	<p>Specifies which character to use for standard U.S. Social Security number (SSN) output delimiters.</p> <p><b>Backslash (\):</b> Uses backward slashes as the delimiter in the SSN. For example, 799\45\6789.</p> <p><b>Dash (-):</b> Uses dashes as the delimiter in the SSN. For example, 799-45-6789.</p> <p><b>Slash (/):</b> Uses forward slashes as the delimiter in the SSN. For example, 799/45/6789.</p> <p><b>None:</b> Does not add a delimiter to the SSN. For example, 799456789.</p> <p><b>Period (.):</b> Uses periods as the delimiter in the SSN. For example, 799.45.6789.</p> <p><b>Space:</b> Uses spaces as the delimiter in the SSN. For example, 799 45 6789.</p>

### Related Topics

- [Column sizing](#)

### 5.4.7.2.9 Date options

Configures standards for date-related data.

Option group	Description
Century Threshold	<p>Indicates whether a two-digit date is considered part of the 20th or 21st century. The default value is 25.</p> <p>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).</p> <p>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.</p>
Date Delimiter	<p>Specifies what character to use for standard date output delimiters.</p> <p><b>Backslash (\):</b> Uses backward slashes as the delimiter for the date. For example, 04\01\2010.</p> <p><b>Dash (-):</b> Uses dashes as the delimiter for the date. For example, 04-01-2010.</p> <p><b>Slash (/):</b> Uses forward slashes as the delimiter for the date. For example, 04/01/2010.</p> <p><b>None:</b> Does not add a delimiter to the date. For example, 04012010</p> <p><b>Period (.):</b> Uses periods as the delimiter for the date. For example, 04.01.2010.</p> <p><b>Space:</b> Uses spaces as the delimiter for the date. For example, 04 01 2010.</p> <p><b>Chinese_Japanese:</b> Uses the following Chinese/Japanese characters as delimiters:</p> <ul style="list-style-type: none"> <li>• 月 always follows the month</li> <li>• 日 always follows the day</li> <li>• 年 always follows the year</li> </ul> <p>An example of Arabic numbers with Chinese/Japanese delimiters is:</p> <p>04月01日2010年</p> <p>An example of Chinese/Japanese Numbers with Chinese/Japanese delimiters is:</p> <p>四月一日二零一十年</p>

Option group	Description
Date Format	<p>Specifies how to standardize date output.</p> <p>YEAR_MONTH_DAY: For Example, 2012-08-16</p> <p>YEAR_DAY_MONTH: For Example, 2012-16-08</p> <p>MONTH_DAY_YEAR: For Example, 08-16-2012</p> <p>DAY_MONTH_YEAR: For Example, 16-08-2012</p>
Enable Zero Pad	<p>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.</p> <p><b>Yes:</b> Turns on the option.</p> <p><b>No:</b> Turns off the option.</p>
Input Month Before Day	<p>Specifies whether the date follows the pattern of having the month first or the day first in the input.</p> <p><b>Yes:</b> The month is first. For example, 11/12/2004 would be November 12, 2004.</p> <p><b>No:</b> The day is first. For example, 11/12/2004 would be December 11, 2004.</p>
Input Year First	<p>Specifies whether the date follows the pattern of having the year first in the input.</p> <p><b>Yes:</b> The year is first. For example, if your input is 03/02/04, the transform will convert it to 2003 February 4.</p> <p><b>No:</b> The month is first. For example, 03/02/04 would be March 2, 2004.</p>
Month Format	<p>Specifies how to standardize date and month components.</p> <p><b>Full_Text:</b> 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.</p> <p><b>Numeric:</b> Standardizes output with numeric months (for example, 03).</p> <p><b>Short_Text:</b> 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.</p>

Option group	Description
Numeric Format	<p>Specifies the format of numeric date values</p> <p><b>Arabic_Numbers:</b> Returns numeric date values in Arabic</p> <p><b>Chinese_Japanese_Numbers:</b> Returns numeric date values in Chinese or Japanese.</p>
Year Format	<p>Specifies how to standardize date and year components.</p> <p><b>Full:</b> Standardizes output with four-digit years (for example, 2004).</p> <p><b>Short:</b> Standardizes output with two-digit years (for example, 04).</p>

#### 5.4.7.2.10 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	<p><b>UDPM:</b> Parses data using user-defined patterns created in Cleansing Package Builder.</p> <p><b>Email:</b> Parses data as an e-mail address.</p> <p><b>SSN:</b> Parses data as a U.S. Social Security number.</p> <p><b>Date:</b> Parses data as a date.</p> <p><b>North American Phone:</b> Parses data as a North American Numbering Plan (NANP) telephone number.</p> <p><b>International Phone:</b> Parses data as an international telephone number. Phone number patterns must be defined in Cleansing Package Builder for countries that do not follow the North American Numbering Plan (NANP) .</p> <p><b>Firm:</b> Parses data as firm name.</p> <p><b>Person:</b> Parses data as a personal name</p> <p><b>Person or Firm:</b> Parses data as a personal or firm name.</p> <p><b>Custom Parser:</b> Parses custom operational or product data for the category in the specified custom cleansing package.</p>

### 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.
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_Content_Domain_Sequence (Dynamic input field)	The content domain sequence. 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.
Option_Output_Format (Dynamic input field)	The format for output 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.
Person1_Family_Name1 Person2_Family_Name1	Discrete family name (for example, Smith).

Name	Description
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 Person2_Given_Name1-2	Discrete given names (for example, John or B.).
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 Topics

- [Dynamic transform settings](#)

### 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 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; the standardized and parsed field types are identical.

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.
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.



Generated field name	Content type	Description
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.
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	<p>The match standard for family names.</p> <p>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.</p> <p>If the dictionary does not have an alias entry, the output field is empty.</p>
Family_Name2	Family_Name 2	Second family name. May be used to output the paternal and maternal family names to separate fields.

Generated field name	Content type	Description
Family_Name2_ Match_Std1-6	Family_Name2_ Match_Std	<p>The match standard for second family names.</p> <p>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.</p> <p>If the dictionary does not have an alias entry, the output field is empty.</p>
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	<p>The match standard for firms. For example, HP is the match standard or alias for Hewlett Packard.</p> <p>If the dictionary does not have an alias entry, the output field is empty.</p>
Firm_Location_ Match_Std1-6	Firm_Location_ Match_Std	<p>The match standard for firm locations. For example, MS is the match standard or alias for mailstop.</p> <p>If the dictionary does not have an alias entry, the output field is empty.</p>

Generated field name	Content type	Description
Gender	None	<p>The gender description. The following output is available:</p> <p><b>Ambiguous:</b> The name does not reliably indicate a gender. The name could be either male or female. For example, Pat.</p> <p><b>Male_Strong:</b> High confidence that the person is male. That is, the name belongs to someone who is almost certainly a male. For example, John.</p> <p><b>Male_Weak:</b> Some confidence that the person is male. That is, the name belongs to someone who is probably male. For example, Terry.</p> <p><b>Female_Strong:</b> High confidence that the person is female. That is, the name belongs to someone who is almost certainly a female. For example, Mary.</p> <p><b>Female_Weak:</b> Some confidence that the person is female. That is, the name belongs to someone who is probably a female. For example, Lynn.</p> <p>For dual names, the following output is also available:</p> <p><b>Multi_Names_Ambiguous:</b> At least one of the names does not reliably indicate a gender. For example, Pat and John.</p> <p><b>Multi_Names_Female:</b> Some or high confidence that both of the names belong to people who are female. For example, Mary and Lynn.</p> <p><b>Multi_Names_Male:</b> Some or high confidence that both of the names belong to people who are male. For example, John and Terry.</p> <p><b>Multi_Names_Mixed:</b> Some or high confidence that one of the names belongs to a person who is female, and the other name belongs to a person who is male. For example, Lynn and John.</p>

Generated field name	Content type	Description
Gender_ID	None	<p>A numeric value that corresponds to the gender description:</p> <p><b>0:</b> Unassigned</p> <p><b>1:</b> Male_Strong</p> <p><b>2:</b> Male_Weak</p> <p><b>3:</b> Ambiguous</p> <p><b>4:</b> Female_Weak</p> <p><b>5:</b> Female_Strong</p> <p><b>6:</b> Multi_Names_Mixed</p> <p><b>7:</b> Multi_Names_Male</p> <p><b>8:</b> Multi_Names_Female</p> <p><b>9:</b> Multi_Names_Ambiguous</p>
Given_Name1	Given_Name	Given name (for example, Robert).
Given_Name1_Match_Std1-6	Given_Name1_Match_Std	<p>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.</p> <p>Match standards can help you overcome two types of matching problems: alternate spellings (Catherine and Katherine) and nicknames (Pat and Patrick).</p>
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.
Honorary_Postname	Postname	Honorary postname indicating certification, academic degree, or affiliation. For example, CPA.

Generated field name	Content type	Description
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.  If the dictionary does not have an alias entry, the output field is empty.
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 city code of an international phone number.
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 dictionary does not have an alias entry, the output field is empty.
Name_Connector	None	The connector component of a dual name. For example, and.
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.

Generated field name	Content type	Description
North_American_Phone_Line	Phone	The last four numbers (excluding an extension) parsed from a phone number. In (608) 555-5555, 5555 is returned.
North_American_Phone_Prefix	Phone	The middle three numbers parsed from a phone number. In (608) 555-5555, 555 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.
Prenome	Prenome	Prenome (for example, Mr.).
Prenome_Match_Std1-6	Prenome_Match_Std	The match standard for a prename. For example, MR. is the match standard or alias for Mister. If the dictionary does not have an alias entry, the output field is empty.
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.
SSN_Serial	SSN	The last four numbers in a Social Security number.
Title	Title	Job or occupational title of a person. For example, Manager.

Generated field name	Content type	Description
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.

### 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.

**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`.

### 5.4.8.1 Common

Option	Description
<b>Run as Separate Process</b>	<b>Yes:</b> Splits the transform into a separate process. <b>No:</b> Keeps the transform in the same process as the rest of the dataflow.

### 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.

Option	Description
<b>Delstats Directory</b>	<code>dsf.dir</code> Type the path and file name of the delivery statistics directory, or use the substitution variable <code>\$\$RefFilesAddressCleanse</code> . SAP BusinessObjects 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).

### 5.4.8.3 Processing Options



Option	Description
<b>Site Location</b>	<p>Indicates at which site the current job was processed (if your company has multiple locations that provide DSF2 walk sequence processing).</p> <p>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.</p>
<b>USPS Certification Testing Mode</b>	<p>Specifies the type of certification you are performing for DSF2 walk sequence if applicable:</p> <p><b>None:</b> 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.</p> <p><b>Invoice:</b> 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.</p> <p><b>Sequence:</b> 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.</p>

#### 5.4.8.4 USPS License Information

The USPS License Information options in the DSF2 Walk Sequencer transform are required:

Option	Description
<b>DSF2 License ID</b>	Enter your DSF2 identification number, as the USPS assigned it to you. You can use the substitution variable \$\$DSF2LicenseeID.
<b>Licensee Name</b>	Enter the name of the DSF2 licensed service provider. You can use the substitution variable \$\$USPSLicenseeName.
<b>List ID</b>	Enter the unique 6-digit identification code that you (the DSF2 licensee) assigned to the customer who owns the list.

#### 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
<b>Replace Null With Space</b>	<p>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.</p> <p><b>Yes:</b> Convert NULL to blank spaces.</p> <p><b>No:</b> Do not convert NULL to blank spaces.</p>
<b>Right Pad With Spaces</b>	<p>Specifies whether to right pad the break key field with spaces.</p> <p>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.</p> <p><b>Yes:</b> Right-pad fields with blank spaces.</p> <p><b>No:</b> Do not right-pad fields with blank spaces.</p> <p><b>Tip:</b> If the <b>Replace NULL with space</b> option is set to <b>Yes</b> and the <b>Right pad with spaces</b> option is set to <b>Yes</b>, then fields with NULL values will be replaced with all spaces on the right (to the length of the field).</p>
<b>Presorted Data</b>	<p>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).</p> <p><b>Yes:</b> The input data has already been presorted by Postcode1 and Sortcode_Route.</p> <p><b>No:</b> The input data has not been sorted yet.</p> <p><b>Tip:</b> Choosing <b>No</b> allows the software to sort your data by Postcode1 and Sortcode_Route. This is the preferred setting for this option.</p>

### 5.4.8.6 Input fields

Field	Description
Delivery_Point	The two-digit DPBC code.
DPV_Status	<p>The DPV status component that is generated for this record.</p> <p><b>D:</b> The primary range is a confirmed delivery point, but the secondary range was not available on input.</p> <p><b>L:</b> The address triggered DPV locking.</p> <p><b>N:</b> The address is not a valid delivery point. The Walk_Sequence_Number output field is 0000.</p> <p><b>S:</b> The primary range is a valid delivery point, but the parsed secondary range is not valid in the DPV directory.</p> <p><b>Y:</b> The address is a confirmed delivery point. The primary range and secondary range (if present) are valid.</p> <p><i>blank:</i> 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.</p>
DSF2_Business_Indicator (optional)	<p>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.)</p> <p><b>Y:</b> Business address.</p> <p><b>N:</b> Not a business address.</p> <p><i>blank:</i> Address was not looked up.</p>
LOT	The Line-of-Travel number.
LOT_Order	<p>The Line-of-Travel sortation:</p> <p><b>A:</b> Ascending</p> <p><b>D:</b> Descending</p> <p>LOT codes are required for non-automated, CART presorting in Standard Mail, Enhanced Carrier Route Subclass.</p>
Postcode1	The five-digit primary ZIP Code. It does not include the four digit ZIP4 Code.

Field	Description
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.

### 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	<p>Indicates whether the postcode1/sortcode route combination qualifies for the 75% total active deliveries discount.</p> <p><b>Y:</b> The postcode1/sortcode route combination qualifies for the 75% total active deliveries discount.</p> <p><b>N:</b> The postcode1/sortcode route combination does not qualify for the 75% total active deliveries discount.</p> <p><i>blank</i>: The record was not sequenced.</p> <p><b>Tip:</b> Active deliveries include residential, business, and PO Box addresses.</p>
Residential_Sat_Discount	<p>Indicates whether the postcode1/sortcode route combination qualifies for the 90% residential saturation discount.</p> <p><b>Y:</b> The postcode1/sortcode route combination qualifies for the 90% residential saturation discount.</p> <p><b>N:</b> The postcode1/sortcode route combination does not qualify for the 90% residential saturation discount.</p> <p><i>blank</i>: The record was not sequenced.</p>

Field	Description
Sortcode_Route_Discount	<p>Indicates whether the postcode1/sortcode route combination qualifies for the Sortcode (Carrier Route) discount.</p> <p><b>Y:</b> The postcode1/sortcode route combination qualifies for the Sortcode discount.</p> <p><b>N:</b> The postcode1/sortcode route combination does not qualify for the Sortcode discount.</p> <p><i>blank</i> : The record was not sequenced.</p> <p><b>Tip:</b> Mailers must have 10 or more deliveries to the same postcode1/sortcode combination to qualify for the discount.</p>
Walk_Sequence_Discount	<p>Indicates whether the postcode1/sortcode route combination qualifies for the walk sequence discount.</p> <p><b>Y:</b> The postcode1/sortcode route combination qualifies for the walk sequence discount</p> <p><b>N:</b> The postcode1/sortcode route combination does not qualify for the walk sequence discount.</p> <p><i>blank</i> : The record was not sequenced.</p> <p><b>Tip:</b> Mailers must have 125 or more sequenced delivery points for each postcode1/sortcode route combination to qualify for the discount.</p>
Walk_Sequence_Number	<p>Indicates the sequence number from 0000 to 9999.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• If Postcode2 field is blank, this field is blank.</li> <li>• If Postcode2 field is not blank and <b>DPV_Status</b> is <b>N</b>, then the field contains 0000.</li> </ul>

## 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 Topics

- [Designer Guide: Data Quality, Address geocoding](#)
- [Designer Guide: Data Quality, Reverse geocoding](#)
- [Designer Guide: Data Quality, POI textual search](#)

### 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 Topics

- [Transform configurations](#)
- [Downloading blueprints and other content objects](#)

## 5.4.9.2 Geocoder options

The Geocoder transform includes options that control how geocoding data is appended to your data.

### 5.4.9.2.1 Common options

Option	Description
Run As Separate Process	<b>Yes:</b> Splits the transform into a separate process. <b>No:</b> Keeps the transform in the same process as the rest of the data flow.

### 5.4.9.2.2 Report and analysis

Use this option to generate report data for the Geocoder transform.

Option	Description
Generate Report Data	<p>Specifies whether to generate report data for this transform.</p> <p><b>Yes:</b> Generates report data for this transform.</p> <p><b>No:</b> Turns off report data generation. If you do not need to generate reports (during testing, for example), you should set this option to <b>No</b> to improve performance.</p> <p><b>Note:</b> 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.</p>



## Related Topics

- [Management Console Guide: Geocoder Summary report](#)

### 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	<p>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.</p> <p><b>Preferred:</b> Assigns to the finest depth. By default, this assigns to the primary number level.</p> <p><b>Primary Number:</b> Assigns to the primary number level.</p> <p><b>Postcode:</b> Assigns to the postcode level.</p> <p><b>Locality:</b> Assigns to the locality, city, or suburb level.</p> <p><b>Smallest Area:</b> 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.</p> <p><b>Note:</b> 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.</p>	Basic, Advanced

Option	Description	Supported directory format								
Distance Unit	<p>Specifies the unit of distance used for the radius.</p> <p><b>Kilometers</b></p> <p><b>Miles</b></p> <p>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.</p>	Advanced								
Locality Assignment Threshold	<p>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.</p> <p>This option is used for address geocoding mode, and also for reverse geocoding with address mode.</p> <p><b>Locality1–4:</b> Returns the locality level that you choose. Locality1 is the most general and Locality4 is the most specific.</p> <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> <p>In this example, there is no Locality4. If you choose Locality4, the finest depth, Locality3, is returned.</p> <p><b>None:</b> Skips the specific assignment level. Use this setting if you do not want to return an assignment threshold on locality.</p> <p><b>Preferred:</b> Assigns to the finest depth at the locality level.</p>	Address	Locality level	Church Cottage	Locality3	Pemborough	Locality2	Bristol	Locality1	Basic, Advanced
Address	Locality level									
Church Cottage	Locality3									
Pemborough	Locality2									
Bristol	Locality1									
Max Records	<p>Specifies the maximum number of records that can be returned. You can enter a number up to 100.</p> <p>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.</p>	Advanced								

Option	Description	Supported directory format
Offset Coordinates	<p>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.</p> <p><b>Yes:</b> Returns the offset values.</p> <p><b>No:</b> Returns the center value regardless of whether the side of the street is known.</p>	Advanced
Postcode Assignment Threshold	<p>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.</p> <p>This option is used for address geocoding mode, and also for reverse geocoding with address mode.</p> <p><b>Postcode Full:</b> 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.</p> <p><b>Postcode1:</b> Assigns to the city or postcode area. For example, in the USA, it assigns to the 5-digit ZIP Code.</p> <p><b>Postcode2 Partial:</b> 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.</p> <p><b>Preferred:</b> Assigns to the finest depth at the postcode level.</p> <p><b>None:</b> Skips the specific assignment level. Use this setting if you do not want to return an assignment threshold on postcode.</p>	Basic, Advanced
Radius	<p>The distance from a specified reference point used to identify an area in which matching records are located.</p> <p>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.</p> <p>In reverse geocoding mode, the maximum radius distance is 111 kilometers or 68.97 miles.</p>	Advanced

### 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, `x:/Business Objects/BusinessObjects/Data Services/DataQuality/reference_data`. If you installed to a different location, you can change the substitution variable while designing the workflow. 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.

#### Related Topics

- [Installation Guide: Additional Information, Directory data](#)

### 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 Topics

- [Input fields](#)
- [Output fields](#)
- [Result\\_List XML output fields](#)
- [Information codes](#)

#### 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	<p>The unit of distance used for the radius. Valid values are:</p> <p><b>Kilometers</b></p> <p><b>Miles</b></p> <p>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.</p> <p>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.</p>	Advanced

Input field name	Category	Description	Supported directory format
Option_Max_Records	Max Records	<p>The maximum number of records that can be returned. You can enter a number up to 100.</p> <p>A value greater than 0 outputs multiple results as XML to the Result_List output field rather than to individual output fields.</p> <p>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.</p> <p>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.</p>	Advanced
Option_Radius	Search Filter	<p>The distance from a specified reference point used to identify an area in which matching records are located.</p> <p>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.</p> <p>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.</p> <p>In reverse geocoding mode, the maximum radius distance is 111 kilometers or 68.97 miles.</p>	Advanced
POI_Name	Address POI	The name of a point of interest, such as the Washington Monument.	Advanced

Input field name	Category	Description	Supported directory format
POI_Type	Address POI	<p>The point-of-interest type expressed as a number; for example, for one vendor 5999 is historical monument.</p> <p>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:</p> <p>8211:8231</p> <p>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.</p>	Advanced
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

Input field name	Category	Description	Supported directory format
Search_Filter_Type	Search Filter	<p>Search criteria for a point-of-interest type, expressed as a four-digit number; for example, for one vendor 5999 is historical monument.</p> <p>If you want to return an address only, enter ADDR.</p> <p>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:</p> <p>8211:8231</p> <p>To return a point-of-interest type and its address, enter:</p> <p>5999:ADDR</p> <p>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.</p>	Advanced

#### 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	<p>The level to which the transform matches the address to the data in the reference fields (directories).</p> <p><b>PRE:</b> 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.</p> <p><b>PRI:</b> Primary Range Interpolated assigns to the level of the address range; for example, 100-500 Main St.</p> <p><b>L1–4:</b> Locality1-4 assigns to the level of city, town, or suburb.</p> <p><b>P1:</b> Postcode1 assigns to the level of Postcode1.</p> <p><b>P2P:</b> Postcode2 Partial assigns the full Postcode1 and the first few characters of Postcode2.</p> <p><b>PF:</b> Postcode Full assigns to the level of Postcode1 and Postcode2, when available.</p>	All	Basic, Advanced
Assignment_Level_Locality	Assignment Level	<p>The level to which the transform assigns the locality.</p> <p><b>L1–4:</b> Returns up to four locality levels. L1 is the most general and L4 is the most specific.</p>	All	Basic, Advanced

Output field name	Category	Description	Supported countries	Supported directory format
Assignment_Level_Postcode	Assignment Level	<p>The level to which the transform assigns the postcode.</p> <p><b>P1:</b> Postcode1 assigns to the level of Postcode1.</p> <p><b>P2P:</b> Postcode2 Partial assigns the full Postcode1 and the first few characters of Postcode2.</p> <p><b>PF:</b> Postcode Full assigns to the level of Postcode1 and Postcode2, when available.</p>	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

Output field name	Category	Description	Supported countries	Supported directory format
Distance	Distance	The distance from the input address, geographical coordinates, or point of interest to the closest address or point of interest.	All	Advanced
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 <a href="#">Information codes</a> .	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

Output field name	Category	Description	Supported countries	Supported directory format
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
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

Output field name	Category	Description	Supported countries	Supported directory format
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 is defined by the government for reporting census information.	USA	Basic, Advanced
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.  <b>0:</b> Undefined. The population may be too large or small to provide accurate data.  <b>1:</b> Over 1 million.  <b>2:</b> 500,000 to 999,999.  <b>3:</b> 100,000 to 499,999.  <b>4:</b> 50,000 to 99,999.  <b>5:</b> 10,000 to 49,999.  <b>6:</b> 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

Output field name	Category	Description	Supported countries	Supported directory format
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
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 Topics

- [Information codes](#)

### 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	<p>The level to which this transform matched the address to the data in the reference fields (directories).</p> <p><b>PRE:</b> 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.</p> <p><b>PRI:</b> Primary Range Interpolated assigns to the level of the address range; for example, 100-500 Main St.</p> <p><b>L1–4:</b> Locality1-4 assigns to the level of city, town, or suburb.</p> <p><b>P1:</b> Postcode1 assigns to the level of Postcode1.</p> <p><b>P2P:</b> Postcode2 Partial assigns the full Postcode1 and the first few characters of Postcode2.</p> <p><b>PF:</b> Postcode Full assigns to the level of Postcode1 and Postcode2, when available.</p>
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.

Output field name	Category	Description
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.
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>

```

#### 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. 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.

Information code	Description
4	<p>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.</p> <p>Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.</p>
5	<p>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.</p> <p>Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.</p>
6	<p>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.</p> <p>For example, if the reference data has two records:</p> <p>Record 1: 100 Main St La Crosse WI 54650</p> <p>Record 2: 100 Main St Bt Micts WI 54650</p> <p>When input with 100 Main ST WI 54650 without a locality name, the 006 information code for ambiguous assignment is output.</p> <p>Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.</p>
7	<p>The input data is blank or invalid.</p> <p>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.</p> <p>Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.</p>
8	<p>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.</p>

Information code	Description
9	<p>The POI type provided on input is invalid. The point-of-interest type is not correct.</p> <p>Verify your input POI type with the POI types described in the appropriate directory update letter.</p>
A	<p>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.</p>
B	<p>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.</p>
C	<p>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.</p> <p>Increase the Max_Record value.</p>
D	<p>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.</p>
E	<p>The closest latitude and longitude to the input address is returned because the input house number does not exist in the geocoder directories.</p> <p>For example:</p> <p>Input address: 100 Main St La Crosse WI 54650</p> <p>Directory data: 1-88 Main St La Crosse WI 54650</p> <p>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.</p>

### Related Topics

- [Output fields](#)

## 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 BusinessObjects 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.

## 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 dataflow.

### 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 Topics

- [Transform configurations](#)
- [Content objects](#)
- [Address Cleanse reference](#)

## 5.4.10.1 Global Address Cleanse transform options

### 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	<p>Specifies whether to generate report data for this transform.</p> <p><b>Yes:</b> Generates report data for this transform.</p> <p><b>No:</b> 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.</p>

#### Related Topics

- [Global Address Cleanse](#)

### 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.

#### Related Topics

- [Installation Guide: Additional Information, Directory data](#)

#### 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	<p>Specifies whether to always use the specified Country Name or to run Country ID processing.</p> <p><b>Constant:</b> 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.</p> <p><b>Assigned:</b> Runs Country ID processing. Choose this option if the input data contains addresses from more than one country.</p>
Country Name	<p>Specifies the country of destination.</p> <p><b>None:</b> Select when the <b>Country ID Mode</b> is set to Assigned and you don't want a default country to be set when the country cannot be identified.</p> <p>Special considerations:</p> <ul style="list-style-type: none"> <li>If the <b>Country ID Mode</b> 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.</li> </ul> <p><b>Note:</b> You cannot choose None if the <b>Country ID Mode</b> option is set to Constant.</p> <ul style="list-style-type: none"> <li>If the <b>Country ID Mode</b> is set to Assigned, choose a country name to be used when the Country ID could not identify a country.</li> <li>If Country Name is set to None, then the address will be sent to the default engine, Global Address.</li> </ul>
Script Code	<p>Specifies the ISO four-character script code for your data.</p> <p><b>CJKK:</b> Chinese, Japanese, and Korean</p> <p><b>GREK:</b> Greek</p> <p><b>LATN:</b> Latin</p> <p><b>None:</b> If this option is selected, transform will attempt to identify the overall script of the input data as CJKK, GREK, LATN, or other.</p>

#### 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.

**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	<p>Specifies how to standardize the address line. (Engines supported: Canada, Global Address, and USA)</p> <p><b>Convert:</b> Converts address lines based on Official address line components instead of Delivery address line components.</p> <p><b>Preserve:</b> Retains non-preferred data in address lines unless the data is incorrect.</p>
Assign Locality	<p>Specifies how to standardize the locality name.</p> <p><b>Convert:</b> Converts the locality name to the locality name preferred by the country's postal authority.</p> <p><b>Preserve:</b> Preserves the input locality name unless it is incorrect.</p> <p><b>Valid:</b> 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.</p>
Capitalization	<p>Specifies the casing of your address data.</p> <p><b>Mixed:</b> Converts data to initial capitals. For example, "MAIN STREET SOUTH" becomes "Main Street South."</p> <p><b>Upper:</b> Converts data to full capitals. For example, "Main Street South" becomes "MAIN STREET SOUTH."</p> <p><b>Note:</b> 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.</p>

Option	Description
Character Width Style	<p>Specifies whether to standardize half-width and full-width characters. This option only applies to Chinese and Japanese data.</p> <p><b>Normal Width:</b> Converts full-width Latin characters to half-width and converts half-width Chinese and Japanese characters to full width.</p> <p><b>Half Width:</b> Convert all characters to half width.</p> <p><b>Full Width:</b> Convert all characters to full width.</p>
Correct As-signed Data	<p>Specifies whether to use the parsed or corrected data for the assigned output fields of type Best.</p> <p><b>Yes:</b> Populates the Best components with corrected data.</p> <p><b>No:</b> Populates the Best components with parsed data.</p> <p><b>Note:</b> If you choose No for this option, the Capitalization option is the only available Standardization option for your assigned data.</p>
Correct Unas-signed Data	<p>Specifies whether the Global Address Cleanse transform standardizes your unassigned data.</p> <p><b>Yes:</b> Populates the Best components with corrected data.</p> <p><b>No:</b> Populates the Best components with parsed data.</p> <p><b>Note:</b> If you choose No for this option, the Capitalization option is the only available Standardization option for your unassigned data.</p>



Option	Description
Country Style	<p>Specifies how to standardize the country data.</p> <p><b>ISO_2CHAR:</b> Standardizes country data to the two-character ISO code, such as AU, CA, or US.</p> <p><b>ISO_3CHAR:</b> Standardizes country data to the three-character ISO code, such as AUS, CAN, or USA</p> <p><b>ISO_3DIGIT:</b> Standardizes country data to the three-digit ISO code, such as 038, 124, or 840.</p> <p><b>Name:</b> Standardizes country data to the full country name, such as Australia, Canada, or United States.</p> <p><b>Preserve:</b> Attempts to retain the country data in the input record, otherwise uses the corrected country value.</p>
Directional Punctuation	<p>Specifies whether to use punctuation in the abbreviated directional data.</p> <p><b>Preserve:</b> If punctuation was provided on input, retains it on output with corrections applied (for example, NW. on input with one period will be N.W. on output with two periods).</p> <p><b>Yes:</b> Outputs directionals with punctuation (for example, N. or S.W.)</p> <p><b>No:</b> Outputs directionals without punctuation (for example, N, SW).</p>
Directional Style	<p>Specifies whether to abbreviate directional data.</p> <p><b>Long:</b> Uses fully-spelled directionals such as "North," "South," "East," "West."</p> <p><b>Preserve:</b> Preserves the style used in the input record.</p> <p><b>Short:</b> Uses abbreviated directionals such as "N," "S," "E," "W."</p>

Option	Description
European Postcode Prefix	<p>Adds the one- to three-character European Postcode prefix, followed by a dash, for mail generated and distributed inside Europe.</p> <p><b>Yes:</b> Adds the European Postcode prefix.</p> <p><b>No:</b> Does not add the European Postcode prefix.</p> <p><b>Preserve:</b> Retains the European Postcode prefix, if one is found on input.</p> <p>In the following address, for example, the D- is the European Postcode extension.</p> <p>Hallesches Ufer 32-38</p> <p>D-10963 Berlin</p> <p>Germany</p> <p><b>Note:</b> The European Postcode prefix is for mail distributed from one European country to another European country.</p>
Extra Lines	<p>Specifies what to do with extra lines of non-address data.</p> <p><b>Preserve:</b> Attempts to retain the extra line of non-address data in the general location in which it was input.</p> <p><b>Remove:</b> Does not include any extra line of non-address data in the standardized lines or multiline fields.</p> <p><b>Preferred:</b> 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.</p>
Format As-signed Data	<p>Specifies whether to format your assigned data based on the country's preferred address format. For example, the format for Germany is:</p> <p>{Primary_Name1} {Primary_Number}</p> <p>{Postcode1} {Locality}</p> <p>{Country}</p> <p><b>Yes:</b> Formats the assigned data.</p> <p><b>No:</b> 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.</p>

Option	Description
Format Unassigned Data	<p>Specifies whether to format your unassigned data based on the country's preferred address format. For example, the format for Germany is:</p> <p>{Primary_Name1} {Primary_Number}</p> <p>{Postcode1} {Locality}</p> <p>{Country}</p> <p><b>Yes:</b> Formats the unassigned data.</p> <p><b>No:</b> Does not format the unassigned data and leaves it in the line in which it was input.</p>
Include Country	<p>Specifies whether to include country names in standardized lines or multiline fields.</p> <p><b>Yes:</b> Includes country name.</p> <p><b>No:</b> Does not include country name.</p> <p><b>Preserve:</b> Retains the country name if found on input.</p>
Include Locality Addition	<p>Specifies whether the Locality1_Full output field contains both the Locality1_Name and the Locality1_Addition information.</p> <p><b>Yes:</b> Includes both the locality and the locality addition information.</p> <p><b>No:</b> Does not include the locality addition.</p> <p><b>Preserve:</b> Includes locality addition if found on input. This is the default setting.</p> <p><b>Note:</b> 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.</p>
Include Unused Address Line Data	<p>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.</p> <p><b>Yes:</b> Outputs the unused address line data in the remainder fields Address_Line_Remainder1 through Address_Line_Remainder4 (for example, 100 Main St Red House).</p> <p><b>No:</b> Does not output the unused address line data (for example, 100 Main St).</p>

Option	Description
Include Unused Lastline Data	<p>Specifies whether to output the unused last line data (for standardized lines and multiline fields):</p> <p><b>Yes:</b> Outputs the unused last line data.</p> <p><b>No:</b> Does not output the unused last line data.</p>
Locality Name Style	<p>Specifies the format for locality data in the Locality1_Name output field for addresses. This option applies to German addresses.</p> <p><b>Preserve:</b> Preserves the locality data format as it was input. This is the default setting.</p> <p><b>Short:</b> Outputs locality data in the abbreviated version, if available in the reference data.</p> <p><b>Note:</b> To use the short locality name style, the <b>Address Line Alias</b> option must be set to <b>Convert</b>.</p> <p><b>Note:</b> This option overrides the Include Locality Addition option.</p>

Option	Description																								
Move Multiline Data	<p>Determines the position of blank lines in output addresses.</p> <p><b>Bottom:</b> If there are any blank lines, the transform moves them to the top and shifts the data to the bottom of the address block.</p> <p><b>No:</b> Does not rearrange any lines, blank or otherwise.</p> <p><b>Top:</b> 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.</p> <p>Example of Top:</p> <div><table><thead><tr><th></th><th>Input data:</th><th></th><th>Result of moving:</th></tr></thead><tbody><tr><td>Line1</td><td>100 Market Street</td><td>↘</td><td>Sycamore Building</td></tr><tr><td>Line2</td><td>Suite 202</td><td>↗</td><td>Suite 202</td></tr><tr><td>Line3</td><td>Sycamore Building</td><td>↘</td><td>100 Market St</td></tr><tr><td>Line4</td><td></td><td>↗</td><td>Boston MA 02109</td></tr><tr><td>Line5</td><td>Boston MA 02109</td><td>↘</td><td></td></tr></tbody></table></div>		Input data:		Result of moving:	Line1	100 Market Street	↘	Sycamore Building	Line2	Suite 202	↗	Suite 202	Line3	Sycamore Building	↘	100 Market St	Line4		↗	Boston MA 02109	Line5	Boston MA 02109	↘	
	Input data:		Result of moving:																						
Line1	100 Market Street	↘	Sycamore Building																						
Line2	Suite 202	↗	Suite 202																						
Line3	Sycamore Building	↘	100 Market St																						
Line4		↗	Boston MA 02109																						
Line5	Boston MA 02109	↘																							
Output Country Language																									

Option	Description
	<p>Specify which language and script to use on output for the country name (not the entire record).</p> <p><b>Preserve:</b> Preserves country name as it was on input.</p> <p><b>Catalan - Latin</b></p> <p><b>Chinese - Hani</b></p> <p><b>Danish - Latin</b></p> <p><b>Dutch - Latin</b></p> <p><b>English - Latin</b></p> <p><b>Finnish - Latin</b></p> <p><b>French - Latin</b></p> <p><b>Greek - Greek</b></p> <p><b>German - Latin</b></p> <p><b>Hungarian - Latin</b></p> <p><b>Italian - Latin</b></p> <p><b>Japanese - Hani</b></p> <p><b>Japanese - Kana</b></p> <p><b>Korean - Hang</b></p> <p><b>Norwegian - Latin</b></p> <p><b>Polish - Latin</b></p> <p><b>Portuguese - Latin</b></p> <p><b>Spanish - Latin</b></p> <p><b>Swedish - Latin</b></p>
Postal Phrase Punctuation	<p>If you choose <b>Short</b> for the Postal Phrase Style option, this option specifies whether to use punctuation in the postal abbreviation.</p> <p><b>Yes:</b> Includes punctuation in postal abbreviations (for example P.O. Box).</p> <p><b>No:</b> Does not insert any punctuation for postal abbreviations (for example, PO Box).</p> <p><b>Preserve:</b> Retains the punctuation of postal abbreviations if found in input record.</p>

Option	Description
Postal Phrase Style	<p>Specifies whether to abbreviate postal phrases.</p> <p><b>Long:</b> Outputs the fully-spelled postal phrase (for example Post Office Box).</p> <p><b>Preserve:</b> Retains the style of the postal phrase if found in the input record.</p> <p><b>Short:</b> Outputs the abbreviated postal phrase (for example, PO Box). The punctuation for this option is determined by the Postal Phrase Punctuation option.</p>
Primary Type Punctuation	<p>If you choose <b>Short</b> for the Primary Type Style option, this option specifies whether to use punctuation in primary type abbreviations.</p> <p><b>Yes:</b> Includes a period at the end of primary type abbreviations (for example, St.).</p> <p><b>No:</b> Does not insert any punctuation at the end of primary type abbreviations (for example, St).</p> <p><b>Preserve:</b> Retains the punctuation of primary type abbreviations from your input.</p>
Primary Type Style	<p>Specifies the style for primary type address elements.</p> <p><b>Long:</b> Uses fully spelled primary types such as Street, Avenue, Road, or Strasse.</p> <p><b>Preserve:</b> Retains the style used in the input record</p> <p><b>Short:</b> Uses abbreviated primary type such as St, Ave, Rd, or Str. The punctuation for this option is determined by the Primary Type Punctuation option.</p>
Region Style	<p>Specifies whether to abbreviate the region name (for example, state or province).</p> <p><b>Long:</b> Uses the fully spelled region name (for example, California or Ontario).</p> <p><b>Preserve:</b> Retains the style used in the input record.</p> <p><b>Short:</b> Abbreviates the region name (for example, CA or ON).</p>
Remove Address Apostrophes	<p>Specifies whether to include apostrophes in certain street names that include a DE L' or D'.</p> <p><b>Yes:</b> Retains apostrophes in street names if it was present on input, for example, Rue D'Abbeville.</p> <p><b>No:</b> Removes apostrophes in street names, for example, Rue D Abbeville.</p>

Option	Description
Secondary Description Punctuation	<p>If you choose <b>Short</b> for the Secondary Description Style, this option specifies whether to use punctuation in the abbreviation.</p> <p><b>Yes:</b> Uses punctuation in the abbreviation (for example, Apt.).</p> <p><b>No:</b> Does not use punctuation (for example, Apt).</p> <p><b>Preserve:</b> Retains the style used in the input record.</p>
Secondary Description Style	<p>Specifies whether to abbreviate the secondary description (for example, a unit or an apartment).</p> <p><b>Long:</b> Uses the fully spelled secondary description (for example, Apartment).</p> <p><b>Preserve:</b> Retains the style used in the input record.</p> <p><b>Short:</b> Abbreviates the secondary description (for example, Apt). The punctuation for this option is determined by the Secondary Description Punctuation option.</p>
Secondary Number Style	<p>Specifies the format of the secondary number (for example, a suite or apartment number). This option applies to Canada and New Zealand addresses.</p> <p><b>Attached:</b> 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.</p> <p><b>Preserve:</b> Preserves the style of the address as it was input.</p> <p><b>Unattached:</b> 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.</p>



Option	Description
Street Name Style	<p>Specifies the format for street data for addresses.</p> <p>This option applies to addresses in Germany and the Netherlands.</p> <p><b>Preserve:</b> Preserves the street data format as it was input.</p> <p>For example:</p> <p>Annelien Kappeyne Van de Coppellostr 2</p> <p>Herten</p> <p>Limburg</p> <p>6049 HD</p> <p><b>Short:</b> 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.</p> <p>For example:</p> <p>A K vd Coppellostr 2</p> <p>Herten</p> <p>Limburg</p> <p>6049 HD</p> <p><b>Note:</b></p> <p>To use the short street name style, the <b>Address Line Alias</b> option must be set to <b>Convert</b>.</p> <p><b>Note:</b></p> <p>The <b>Capitalization</b> option in the Global Address Cleanse transform overrides the <b>Street Name Style</b> option.</p>
Use Local Primary Type Style	<p>Specifies whether to use the type style for primary address components that is present in the address data. Setting this option to <b>Yes</b> ignores the <b>Primary Type Style</b> option. This option applies to Austria, Germany, and Switzerland.</p> <p><b>Yes:</b> Uses the Primary Type Style that is present in the address data.</p> <p><b>No:</b> Uses the Primary Type Style specified in the Primary Type Style option.</p>

Option	Description
Use Postal Country Name	<p>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.</p> <p>If the country does not have a postal country, this option does not change the output.</p> <p><b>Yes:</b> Uses the postal country name.</p> <p><b>No:</b> Uses the territory country name.</p>

#### Related Topics

- [Canada engine](#)
- [Global Address engine options](#)
- [USA engine](#)

#### 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.

**Yes:** Activates the engine for this transform.

**No:** De-activates the engine for this transform.

Option	Description
Engines	<p>Specify which engine to use with this transform.</p> <p><b>Canada</b></p> <p><b>Global Address</b></p> <p><b>USA</b></p>

#### Related Topics

- [Canada engine](#)

- [Global Address engine options](#)
- [USA engine](#)

### *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 Topics**

- [Reference files](#)
- [Canada engine options](#)
- [Canada engine Suggestion List options](#)
- [Canada engine Report options](#)

### *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	<p><b>Yes:</b> Enables non-certified processing of Canada addresses and allows processing with non-POC (Point-of-Call) directories for non-mailing purposes. When you select <b>Yes</b>, you cannot print the SERP Report. Any list created with certification disabled cannot be used for mailing.</p> <p><b>No:</b> Enables certified processing for Canada using POC (Point-of-Call) directories and enables printing of the SERP report.</p>
Dual Address	<p>Specifies the action to take when the Canada engine encounters a dual address.</p> <p><b>Position:</b> 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.)</p> <p><b>Postal:</b> Tries to validate based on the postal address. If that fails, tries again based on the street address.</p> <p><b>Street:</b> Tries to validate based on the street address. If that fails, tries again based on the postal address (rural route or PO box).</p>

Option	Description
Enable LVR Rule	<p>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.</p> <p><b>Yes:</b> 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.)</p> <p><b>No:</b> Disables this rule. The transform reports an LVR address as unassigned when the address line is flawed.</p>
Enable Rural Rule	<p>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.</p> <p>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.</p> <p><b>Yes:</b> 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).</p> <p><b>No:</b> Reports a rural address as invalid if the address line is bad.</p>
Output Address Language	<p><b>Convert:</b> Uses French for records in Quebec, and English for records in other regions (provinces).</p> <p><b>English:</b> Converts records to English.</p> <p><b>French:</b> Converts records to French.</p> <p><b>Preserve:</b> Detects the input language and preserves that language upon output, no matter the region (province).</p>
Parse Only	<p><b>Yes:</b> Parses records into discrete components, but does not perform a lookup in the postal directories. Parse Only is fast, but parsing results are unverified.</p> <p><b>No:</b> 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 data is available.</p>

Option	Description
Postcode No Match Search	<p>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:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> </ul> <p>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.</p> <p><b>Yes:</b> Turns on this option. (This value is necessary for SERP certification.)</p> <p><b>No:</b> Turns off this option.</p>
Postcode Only Search	<p>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.</p> <p><b>Yes:</b> Turns on the option. (This value is necessary for SERP certification.)</p> <p><b>No:</b> 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.</p>
Postcode Priority Over Street	<p>This option is important when the Canada engine is trying to break a tie between two possible assignments:</p> <ul style="list-style-type: none"> <li>• A near match on address line</li> <li>• An exact match on postal code</li> </ul> <p><b>Yes:</b> When breaking a tie between a near match on address line and an exact match on postal code, validates based on the postal code. (This value is necessary for SERP certification.)</p> <p><b>No:</b> When breaking a tie between a near match for the address line and an exact match for the postal code, places more weight on the address line than on the postal code, because data-entry errors are common in postal codes. Where possible, the transform changes the postal code to agree with the address line.</p>

Option	Description
Unit Description	<p>Specifies the unit description in English:</p> <p><b>Apartment:</b> Uses Apartment as the default unit designator.</p> <p><b>Default:</b> Uses the default unit designator.</p> <p><b>Unit:</b> Uses Unit as the default unit designator.</p>

### Related Topics

- [Standardization options](#)

### 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).

### 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	<p>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.</p> <p>Type a value from 0 to 80.</p>
Address Range	<p>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:</p> <p>1000 Pine St.</p> <p>Suggestions would be returned in a range from 750 to 1250 Pine Street.</p> <p>If you don't want to limit the ranges returned in suggestions, type 0.</p>
Combine Overlapping Ranges	<p>Specifies whether individual suggestions with overlapping ranges are combined.</p> <p><b>Yes:</b> Ignores gaps and overlaps in ranges.</p> <p>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.</p> <p>For example, the following suggestions would be presented if this option is set to No:</p> <p>1000-1099 Maple Ave</p> <p>1100-1199 Maple Ave</p> <p>But this suggestion would only show if set to Yes:</p> <p>1000-1199 Maple Ave</p> <p><b>No:</b> Does not combine overlapping ranges.</p>
Enable Suggestion Lists	<p>Specifies whether suggestion lists are generated.</p> <p><b>No:</b> Does not generate suggestion lists.</p> <p><b>Yes:</b> Generates suggestion lists when assignment candidates are present.</p>

Option	Description
Lastlines Match Minimum	<p>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.</p> <p>Type a value from 0 to 80.</p>
Max Number Address Lines	<p>Specifies the maximum number of address line suggestions that can be generated.</p> <p>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.</p> <p>The minimum number you can enter is 2. The maximum number you can enter is 50.</p>
Max Number Lastlines	<p>Specifies the maximum number of lastline suggestions that can be generated.</p> <p>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.</p> <p>The minimum number you can enter is 2. The maximum number you can enter is 15.</p>

### *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.



Option	Description
Disable Certification	<p>Specifies whether to perform non-certified or certified processing of addresses for Australia, France, or New Zealand.</p> <p><b>Note:</b></p> <p>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 <b>No</b> 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.</p> <p><b>Australia:</b></p> <ul style="list-style-type: none"> <li>• <b>Yes:</b> 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.</li> </ul> <p>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.</p> <ul style="list-style-type: none"> <li>• <b>No:</b> Uses the most current directory information, disables non-certified features, and enables printing of the AMAS report.</li> </ul> <p><b>France:</b></p> <ul style="list-style-type: none"> <li>• <b>Yes:</b> Enables non-certified processing of France addresses for non-mailing purposes.</li> <li>• <b>No:</b> 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.</li> </ul> <p><b>Note:</b></p> <p>To return the address in all upper case, set the Capitalization option under <b>Standardization Options &gt; Country &gt; Options to Upper</b>.</p> <p><b>New Zealand:</b></p> <ul style="list-style-type: none"> <li>• <b>Yes:</b> Enables non-certified processing of New Zealand addresses and allows processing with expired directories for non-mailing purposes.</li> </ul> <p>When you select Yes, you cannot print the SOA Report. Any list created with certification disabled cannot be used for mailing.</p> <ul style="list-style-type: none"> <li>• <b>No:</b> Enables certified processing for New Zealand and enables printing of the SOA report.</li> </ul>

Option	Description
Dual Address	<p>Specifies the action to take when the Global Address engine encounters a dual address.</p> <p><b>Position:</b> 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.</p> <p><b>Postal:</b> Attempts to validate based on the postal address. If that fails, attempts again based on the street address.</p> <p><b>Street:</b> Attempts to validate based on the street address. If that fails, attempts again based on the postal address.</p>
Retain Postcode if Valid Format	<p>Set this option for processing valid or invalid input postcodes.</p> <p><b>Yes:</b> Retains input postcode unless it is an invalid format for the country and there is a single answer or intelligent matching is possible.</p> <p><b>No:</b> Updates output postcode if there is a single answer or intelligent matching is possible. Otherwise retain input postcode.</p> <p><b>Note:</b> If the postcode is blank, the transform will add one if there is a single answer.</p>

### *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	<p>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:</p> <p>1000 Pine St.</p> <p>Suggestions would be returned in a range from 750 to 1250 Pine Street.</p> <p>If you don't want to limit the ranges returned in suggestions, type 0. The maximum value is 5000.</p>

Option	Description
Enable Suggestion Lists	<p>Specifies whether suggestion lists are generated.</p> <p><b>No:</b> Does not generate suggestion lists.</p> <p><b>Yes:</b> Generates suggestion lists when assignment candidates are present.</p>
Max Number Address Lines	<p>Specifies the maximum number of address-line suggestions that can be generated.</p> <p>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.</p> <p>The minimum number you can enter is 2. The maximum number you can enter is 200. The default setting is 100.</p>
Max Number Lastlines	<p>Specifies the maximum number of lastline suggestions that can be generated.</p> <p>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.</p> <p>The minimum number you can enter is 2. The maximum number you can enter is 100. The default setting is 60.</p>

### *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](#)
- [Report options for Australia](#)

### *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).

Option	Description
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).
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.
SOA Issuer Name	Specifies the name of the company that prepared this list (up to 40 characters).

### *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).

Option	Description
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).

### *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 Topics**

- [USA engine Suggestion List options](#)
- [Standardization options](#)

### *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	<p>Specifies the action to take when the transform encounters a dual address.</p> <p><b>Position:</b> Selects an address based on the arrangement of the input data.</p> <p>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.</p> <p><b>Postal:</b> Attempts to validate based on the postal address. If that fails, attempts again based on the street address.</p> <p><b>Street:</b> Attempts to validate based on the street address. If that fails, attempts again based on the postal address (rural route or PO Box).</p>

Option	Description
Parse Only	<p><b>Yes:</b> Parses records into discrete components, but does not perform a lookup in the postal directories. Parse Only is fast, but parsing results are unverified.</p> <p><b>No:</b> 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.</p>
Unit Description	<p>Specifies how to standardize the unit description.</p> <p><b>Convert:</b> Uses the unit description found in the postal directory (such as an apartment, suite, room, or floor).</p> <p><b>Preserve:</b> Preserves the unit description from the input record, correcting any spelling errors.</p>

### Related Topics

- [Standardization options](#)

### USA engine Suggestion List options

Set these options to generate suggestion lists for the USA and its territories.

Option	Description
Address Lines Match Minimum	<p>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.</p> <p>Type a value from 0 to 80.</p>
Address Range	<p>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:</p> <p>1000 Pine St.</p> <p>Suggestions would only be returned in a range from 750 to 1250 Pine Street.</p> <p>If you don't want to limit the ranges returned in suggestions, type 0.</p>

Option	Description
Combine Overlapping Ranges	<p>Specifies whether individual suggestions with overlapping ranges are combined.</p> <p><b>Yes:</b> Ignores gaps and overlaps in ranges.</p> <p>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.</p> <p>For example, a suggestion list might show the following suggestions if this option is set to No:</p> <p>1000-1099 Maple Ave</p> <p>1100-1199 Maple Ave</p> <p>But would only show this suggestion if set to Yes:</p> <p>1000-1199 Maple Ave</p> <p><b>No:</b> Does not combine overlapping ranges.</p>
Enable Suggestion Lists	<p>Specifies whether suggestion lists are generated.</p> <p><b>Yes:</b> Generates suggestion lists when assignment candidates are present.</p> <p><b>No:</b> Does not generate suggestion lists.</p>
Lastlines Match Minimum	<p>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.</p> <p>Type a value from 0 to 80.</p>
Max Number Address Lines	<p>Specifies the maximum number of address line suggestions that can be generated.</p> <p>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.</p> <p>The minimum number you can enter is 2. The maximum number you can enter is 100.</p>

Option	Description
Max Number Lastlines	<p>Specifies the maximum number of lastline suggestions that can be generated.</p> <p>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.</p> <p>The minimum number you can enter is 2. The maximum number you can enter is 100.</p>

#### 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 5-112: Suggestion List group

Option	Description
Suggestion List Components	Choose the address component fields that you want to include in the address suggestions.
Delimiter	<p>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.</p> <p>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.</p>
Field Delimiter	<p>Specifies a character to use to separate each field in a single suggestion list.</p> <p>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.</p> <p>This value can be any character string. The default value is a pipe symbol ( ). This value should differ from the Delimiter value.</p>



Option	Description
Output Style	<p>Specifies the format for how the suggestion list data should be output.</p> <p><b>Delimited:</b> Outputs the suggestion list data in a delimited text format, with the delimiters specified in the Delimiter and Field Delimiter options.</p> <p><b>XML:</b> 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.</p>

Table 5-113: Suggestion List components

Field	Description
Building_Name	<p>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.”</p> <p>In some cases, an address will include the bulding name and primary number.</p>
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”).

Field	Description
Delivery_Installation_Type	<p>The delivery installation type.</p> <p><b>English:</b></p> <p><b>PO:</b> Post Office.</p> <p><b>RPO:</b> Retail Post Outlet.</p> <p><b>STN:</b> Station.</p> <p><b>LCD:</b> Letter Carrier Depot.</p> <p><b>CMC:</b> Community Mail Center.</p> <p><b>CDO:</b> Commercial Dealership Outlet.</p> <p><b>French:</b></p> <p><b>BDP:</b> Bureau de Poste.</p> <p><b>CSP:</b> Comptoir Service Postal.</p> <p><b>SUCC:</b> Succursale.</p> <p><b>PDF:</b> Poste de Facteurs.</p> <p><b>CPC:</b> Centre Postal Communautaire.</p> <p><b>CC:</b> Concession Commerciale.</p>
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 Locality3_Official Locality4_Official	The locality name preferred by the postal authority.

Field	Description
Postcode	Returns the postal code.
Postcode1	<b>Australia:</b> Four-digit postcode. <b>Canada:</b> First three characters (FSA) of the postal code. <b>Global:</b> Postal code. <b>USA:</b> Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4).
Postcode2	The secondary postal code. <b>Canada:</b> The last three characters (LDU) of the postal code. <b>USA:</b> 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").

Field	Description
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 Street and PO box.  <b>E:</b> The record is even-numbered.  <b>O:</b> The record is odd-numbered.  <b>B:</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	Returns the state, province, or region.
Secondary_Side_Indicator	Indicates if even, odd, or both values are valid. This applies to floors and units.  <b>E:</b> The secondary record is even-numbered.  <b>O:</b> The secondary record is odd-numbered.  <b>B:</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."

Field	Description
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.

### 5.4.10.2 Global Address Cleanse fields

The Global Address Cleanse transform requires that you map fields on input and output.

#### 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.

Category	Description
Field Addrclass	<p>Specifies the address class for the generated field.</p> <p><b>Delivery:</b> 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.</p> <p><b>Dual:</b> 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.</p> <p><b>Official:</b> When used with the applicable Field Name, this value generates fields in the form of the data preferred by the Postal Authority.</p> <p>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.</p>
Field Category	<p><b>Component:</b> Individual address components and postal codes that are related to the processed record.</p> <p><b>Standardized:</b> Standardized input lines based on the settings in the Standardization Options group in the transform.</p> <p><b>Suggestion:</b> Suggestion list output data based on the settings in the Suggestion List Options group.</p>

Category	Description
Field Class (USA Regulatory Address Cleanse)	<p>Specifies the field class that you want to assign to your output fields.</p> <p><b>Best:</b> 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.</p> <p><b>Note:</b> 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.</p> <p><b>Correct:</b> 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.</p> <p><b>Parsed:</b> Outputs the parsed value.</p> <p><b>Pre_LACSLink:</b> Retained address components that were replaced with LACSLink address information.</p> <p><b>Move_Updated:</b> Outputs the address components that have been updated with move-updated address data.</p> <p><b>Note:</b> 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.</p>
Field Class (Global Address Cleanse)	<p>Specifies the field class that you want to assign to your output fields.</p> <p><b>Best:</b> 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.</p> <p><b>Parsed:</b> Outputs the parsed value.</p>
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 Topics**

- [Content types](#)
- [Designer Guide: How address cleanse works](#)

**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	<p>The delivery address line, for example, "123 Main Street, Unit 4."</p> <p><b>Japan:</b> Address_Line may represent the following address components:</p> <ul style="list-style-type: none"> <li>• Block (chome, kumi, Hokkaido go), sub-block (banchi, gaiku, tochi kukaku), and house number (go) parts of the Japanese address.</li> <li>• The Building Name, Building Floor, Building Room parts of the Japanese address.</li> <li>• The P.O. Box portion of the address, if applicable.</li> </ul> <p><b>China:</b> Address_Line may represent the following address components:</p> <ul style="list-style-type: none"> <li>• Street and street number.</li> <li>• Building, Floor, Unit</li> <li>• Residential community</li> </ul> <p>For example,</p> <p>晨晖路123号中华大厦12楼1201室</p> <p>宝山新村100号201室</p>	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	<p>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.</p> <p><b>Japan:</b> All Firm data for addresses in Japan should be placed in this field.</p> <p><b>China:</b> 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.</p>	All engines
Lastline	The locality, region (when it is included in the data), and postal code on one line.	All engines
Locality1	<p>The city, town, or suburb.</p> <p><b>Japan:</b> The city (shi), island (shima), ward (ku), county (gun) district (machi) or village (mura).</p> <p><b>China:</b> 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 (省直辖区 shengzhixixian).</p>	All engines
Locality2	<p>Any additional city, town, or suburb information.</p> <p><b>USA:</b> The Puerto Rican urbanization.</p> <p><b>Japan:</b> Any additional ward, district, village or sub-district (aza, bu, chiwari, sen)</p> <p><b>China:</b> 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).</p>	G, U

Field	Description	Engine(s)
Locality3	<p>Any additional city, town, or suburb information.</p> <p><b>Japan:</b> Any additional district, village, sub-district (aza, bu, chiwari, sen, donchi, and tori), or super block (joh).</p> <p><b>China:</b> 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).</p>	G
Multiline1-12	<p>A line that may contain any data. The type of data in this line may vary from record to record.</p> <p><b>Japan:</b> 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.</p>	All engines
Postcode	<p>The postal code.</p> <p><b>USA:</b> The five-digit ZIP Code and ZIP+4.</p>	All engines
Region1	<p>The state, province, or region.</p> <p><b>Japan:</b> Represents the prefecture (to, do, fu, ken). A prefecture is similar to a state in the U.S.</p> <p><b>China:</b> Province-level regions, Provinces (省 sheng), Autonomous regions (自治区 zizhiqu), Municipalities (直辖市 zhixiashi), Special administrative regions (特别行政区 tebie xingzhengqu).</p>	All engines
Suggestion_Reply1-6	<p>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.</p> <p><b>Suggestion_Reply1:</b> If you do not want to use a suggestion list, make the value of this field 0 and the suggestion list will be ignored.</p>	All engines

### 5.4.10.2.3 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)

Field	Description	Engine
Additional_Info1	<p><b>Austria:</b> Includes the PAC code of the currently valid address when you choose to preserve the alias address on output.</p> <p><b>Belgium:</b> Includes the NIS code.</p> <p><b>Canada:</b> The official 13-character abbreviation of the city name, or the full spelling if the city name is less than 13 characters (including spaces).</p> <p><b>France:</b> Includes the INSEE code.</p> <p><b>Germany:</b> Includes a portion of the German freightcode (Frachtleitcode).</p> <p><b>Liechtenstein:</b> Includes the postal service district (Botenbezirke) when it is available in the data.</p> <p><b>Poland:</b> Includes the district name (powiat).</p> <p><b>Spain:</b> Includes the INE 91 section code.</p> <p><b>Switzerland:</b> Includes the postal service district (Botenbezirke) when it is available in the data.</p>	C, G

Field	Description	Engine
Additional_Info2	<p><b>Austria:</b> Includes the City ID (OKZ).</p> <p><b>Canada:</b> The official 18-character abbreviation of the city name, or the full spelling if the city name is less than 18 characters (including spaces).</p> <p><b>Germany:</b> Includes the District Code.</p> <p><b>Liechtenstein:</b> Additional postcode.</p> <p><b>Poland:</b> Includes the community name (gmina).</p> <p><b>Spain:</b> Includes the INE Street code.</p> <p><b>Switzerland:</b> Additional postcode.</p>	C, G
Additional_Info3	<p><b>Austria:</b> Includes the Pusher-Leitcode (parcel).</p> <p><b>Germany:</b> Includes the German City ID (ALORT).</p> <p><b>Spain:</b> Includes the INE Town code.</p>	G
Additional_Info4	<p><b>Austria:</b> Includes the Pusher-Leitcode (letter).</p> <p><b>Germany:</b> Includes the German street name ID (StrSchl).</p>	G
Additional_Info5	<p><b>Austria:</b> Includes the SKZ Street Code (7-digit).</p> <p><b>Germany:</b> Includes the discount code for the freightcode.</p>	G
Additional_Info6	<p><b>Austria:</b> Includes the corner-house identification (1-digit). The value for a corner house is <b>1</b>.</p>	G
Additional_Info7-8	Reserved for future use.	All engines
Address_Line_Remainder1-4	<p>Extraneous data found in the address line, which either cannot be identified or does not belong in a standardized address.</p> <p>USA 1-2: Complete secondary non-postal address (for example, Apt. 10, Ste 500, Box 34, Rm 7, 5th Flr).</p>	All engines

Field	Description	Engine
Address_Type	A one-character code that represents the type of address identified:  <b>P</b> : Postal <b>S</b> : Street <b>X</b> : Unknown	All engines
Area_Name1	An industrial area such as RIICO INDUSTRIAL AREA.	G
Assignment_Level	The level to which this transform matched the address to the data in the reference files (directories):  <b>C</b> : Country <b>L1</b> : Locality1 <b>L2</b> : Locality2 <b>L3</b> : Locality3 <b>L4</b> : Locality4 <b>PN</b> : Primary name <b>PR</b> : Primary range <b>R</b> : Region <b>S</b> : Secondary <b>X</b> : Unknown, or the address was unassigned	All engines

Field	Description	Engine
Assignment_Type	<p>A one- or two-character code that represents the type of address.</p> <p>Engine support varies; see each code listing for supported engines.</p> <p><b>BN:</b> Building name (Canada, Global Address)</p> <p><b>F:</b> Firm (Canada, Global Address, USA)</p> <p><b>G:</b> General delivery (Canada, Global Address, USA)</p> <p><b>H:</b> High-rise building (Canada, USA)</p> <p><b>HB:</b> House Boat (Global Address)</p> <p><b>L:</b> LOT (Global Address)</p> <p><b>M:</b> Military (Canada, USA)</p> <p><b>R:</b> Rural (Canada, USA)</p> <p><b>P:</b> Postal (Canada, Global Address, USA)</p> <p><b>PR:</b> Poste Restante (Global Address)</p> <p><b>PS:</b> Packstation or Paketbox (Global Address)</p> <p><b>RP:</b> Postal Served by Route (Global Address)</p> <p><b>S:</b> Street (Canada, Global Address, USA)</p> <p><b>SR:</b> Street served by route (Canada, Global Address)</p> <p><b>U:</b> Uninhabited (Global Address)</p> <p><b>W:</b> Caravan (Global Address)</p> <p><b>X:</b> Unknown or the address was unassigned (Canada, Global Address, USA)</p>	All engines
Block_Description	Block description such as "Block."	G
Block_Number	Block number.	G

Field	Description	Engine
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
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. <b>USA:</b> 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. <b>Japan:</b> Returns the post office name.	C, G
Delivery_Installation_Qualifier	Delivery Installation qualifier (for example, “Main” in “RR 2 Vancouver Stn Main”).	C

Field	Description	Engine
Delivery_Installation_Type	<p>The delivery installation type.</p> <p><b>English:</b></p> <p><b>PO:</b> Post Office.</p> <p><b>RPO:</b> Retail Post Outlet.</p> <p><b>STN:</b> Station.</p> <p><b>LCD:</b> Letter Carrier Depot.</p> <p><b>CMC:</b> Community Mail Center.</p> <p><b>CDO:</b> Commercial Dealership Outlet.</p> <p><b>French:</b></p> <p><b>BDP:</b> Bureau de Poste.</p> <p><b>CSP:</b> Comptoir Service Postal.</p> <p><b>SUCC:</b> Succursale.</p> <p><b>PDF:</b> Poste de Facteurs.</p> <p><b>CPC:</b> Centre Postal Communautaire.</p> <p><b>CC:</b> Concession Commerciale.</p>	C
Delivery_Point	<p><b>Australia:</b> Eight-digit delivery point identifier. This is the primary component needed to generate a barcode.</p> <p>This component is not printed on mail pieces.</p> <p><b>Austria:</b> Includes the PAC code, which is a unique identifier assigned by the Austrian postal authority.</p> <p><b>New Zealand:</b> A seven-character code that represents the delivery-point identifier.</p> <p><b>United Kingdom:</b> A two-character code that represents the delivery-point suffix.</p>	G
Engine_Name	The name of the engine that was selected to process the record.	All engines



Field	Description	Engine
Error	<p>Specifies the error status generated as the result of looking up the current record and performing suggestion processing. Possible output values are 0- 5.</p> <p><b>0</b> No suggestion selection error.</p> <p><b>1</b> Blank suggestion selection/entry.</p> <p><b>2</b> Invalid suggestion selection.</p> <p><b>3</b> Invalid primary range.</p> <p><b>4</b> Invalid floor range.</p> <p><b>5</b> Invalid unit range.</p>	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
Firm	<p>The firm name for the address.</p> <p>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.</p> <p><b>Canada and USA:</b> 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.</p> <p><b>Global Address:</b> If the firm name is available on input, the Global Address engine returns the firm name.</p>	All engines
Floor_Description	<p>The level description, such as "Floor."</p> <p><b>Japan:</b> The level description, such as kai.</p>	G
Floor_Number	The level number or information.	G

Field	Description	Engine
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	<p>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.</p> <p>For more information, see <a href="#">Information codes (Global Address Cleanse)</a>.</p>	All engines
ISO_Country_Code_2Char	The two-character ISO code that identifies the country, for example, DE is Germany.	All engines
ISO_Country_Code_3Char	The ISO-3166 three-character code that identifies the country, for example, DEU is Germany.	All engines
ISO_Country_Code_3Digit	The three-digit ISO code that identifies the country, for example, 276 is Germany.	All engines
Language	The 2-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

Field	Description	Engine
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. <b>Japan:</b> Locality1 descriptor. For example, shi, shima, and so on. <b>China:</b> 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. <b>Japan:</b> The city (shi), island (shima), ward (ku), county (gun), district (machi), or village (mura).	All engines
Locality2_Name	Additional locality information. <b>USA:</b> Urbanization (Puerto Rican addresses only).	G, U

Field	Description	Engine
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
Match_Block_Number	Reserved for future use.	All engines
Match_Floor_Number	Reserved for future use.	All engines
Match_Locality	Reserved for future use.	All engines
Match_Primary_Directional	Reserved for future use.	All engines
Match_Primary_Name	Reserved for future use.	All engines
Match_Primary_Number	Reserved for future use.	All engines
Match_Primary_Type	Reserved for future use.	All engines
Match_Stairwell	Reserved for future use.	All engines
Match_Unit_Number	Reserved for future use.	All engines
Match_Wing_Name	Reserved for future use.	All engines

Field	Description	Engine
Multiline1-12	A line that may contain any data. The type of data in this line may vary from record to record.	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
Postcode_Full	<p><b>Australia:</b> Complete four-digit postal code.</p> <p><b>Canada:</b> Complete six-character postal code (FSA + LDU).</p> <p><b>Global Address:</b> Complete postal code.</p> <p><b>USA:</b> The full ZIP Code with a hyphen (10 characters).</p> <p><b>Japan:</b> The seven-digit postal code.</p>	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 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.	G
Postcode1	<p><b>Australia:</b> Four-digit postcode.</p> <p><b>Canada:</b> First three characters (FSA) of the postal code.</p> <p><b>Global Address:</b> Postal code.</p> <p><b>USA:</b> Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4).</p> <p><b>Japan:</b> The first three digits of the postal code.</p>	All engines

Field	Description	Engine
Postcode2	<p>The secondary postal code.</p> <p><b>Canada:</b> The last three characters (LDU) of the postal code.</p> <p><b>USA:</b> 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).</p> <p><b>Japan:</b> Contains the last four digits of the postal code.</p>	All engines
Postcode_Description	<p>A word that indicates a postal code, when available on input. For example:</p> <p><b>Brazil:</b> CEP, which stands for Código de Endereçamento Postal, and is output as CEP 52041-970.</p> <p><b>China:</b> 邮编</p> <p><b>Japan:</b> 〒</p>	G
Post_Office_Name	The name or numeric representation for a post office, such as, "01" BP 1012.	G
Primary_Address	<p>Primary address line, such as the street address or post office box. Does not include secondary address information such as apartment.</p> <p><b>Japan:</b> The full block data.</p>	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	<p>The street name description (typically a street name or box description).</p> <p><b>Japan:</b> Block (chome, kumi, Hokkaido go), sub-block (banchi, gaiku, tochi kukaku).</p> <p>The Post office name description (yuubinnyoku or siten).</p>	All engines

Field	Description	Engine
Primary_Name2	Second street and name description, typically a street name or box description. <b>Japan:</b> Additional block and sub-block information.	G
Primary_Name3	The street name, delivery mode, and so on. <b>Japan:</b> Additional block and sub-block information.	G
Primary_Name4	The street name, delivery mode, and so on. <b>Japan:</b> 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. <b>Japan:</b> The postal number identifier 号 (go) or house number description 号 (go). <b>China:</b> 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. <b>Japan:</b> The postal box identifier.	G
Primary_Number_Full	The primary number, primary number description, and primary number extra.	All engines

Field	Description	Engine
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. <b>Japan:</b> 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. <b>Japan:</b> 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 non abbreviated is based on the standardization setting for Directional Style. <b>Japan:</b> Directional that precedes block or sub-block.	G, U
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. <b>Japan:</b> 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



Field	Description	Engine
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 <a href="#">Quality codes (Global Address Cleanse)</a> .	All engines
Region1	Either the Region1_Name or Region1_Symbol based on the standardization option Region Style.	All engines
Region1_Description	Region1 description.	G
Region1_Full	Includes Region1 and Region1_Description.	All engines
Region1_Name	The fully spelled out Region1 name.	All engines
Region1_Symbol	The two- or three-character representation of the region1 name.	All engines
Region2	Either the Region2_Name or Region2_Symbol based on the standardization option Region Style.	G
Region2_Description	Region2 Description.	G
Region2_Full	Includes Region2 and Region2_Description.	G
Region2_Name	The fully spelled out Region2 name.	G
Region2_Symbol	The two- or three-character representation of the region2 name.	G
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

Field	Description	Engine
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	<p>Specifies the suggestion status generated as the result of looking up the current record and performing suggestion processing.</p> <p><b>A:</b> Primary address-line suggestions available.</p> <p><b>AM:</b> Follow up primary address-line suggestions available.</p> <p><b>F:</b> Floor range is invalid.</p> <p><b>L:</b> Lastline suggestions available.</p> <p><b>N:</b> No suggestions available.</p> <p><b>R:</b> Primary range is invalid.</p> <p><b>S:</b> Unit range is invalid.</p> <p><b>U:</b> Secondary address-line suggestions available.</p> <p><b>UM:</b> Follow up secondary address-line suggestions available.</p>	All engines
Status_Code	<p>Displays a six-character code that always starts with an S. This code explains what parts of the address changed during processing.</p> <p>For more information, see <a href="#">Status codes (Global Address Cleanse)</a>.</p>	All engines
Unit_Description	<p>The unit description, such as "Apartment" or "Flat."</p> <p><b>Japan:</b> The unit description, such as gousitsu.</p>	All engines
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

Field	Description	Engine
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 Topics

- [Information codes \(Global Address Cleanse\)](#)
- [Quality codes \(Global Address Cleanse\)](#)
- [Status codes \(Global Address Cleanse\)](#)

#### 5.4.10.2.4 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](#)
- [Suggestion List Output Fields](#)

#### *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: <ul style="list-style-type: none"> <li>• Lastline</li> <li>• Primary Address</li> <li>• Follow-up Primary Address</li> <li>• Secondary Address</li> <li>• Follow-up Secondary Address</li> </ul>

#### *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	<p>The delivery installation type.</p> <p><b>English:</b></p> <ul style="list-style-type: none"> <li>• <b>PO:</b> Post Office.</li> <li>• <b>RPO:</b> Retail Post Outlet.</li> <li>• <b>STN:</b> Station.</li> <li>• <b>LCD:</b> Letter Carrier Depot.</li> <li>• <b>CMC:</b> Community Mail Center.</li> <li>• <b>CDO:</b> Commercial Dealership Outlet.</li> </ul> <p><b>French:</b></p> <ul style="list-style-type: none"> <li>• <b>BDP:</b> Bureau de Poste.</li> <li>• <b>CSP:</b> Comptoir Service Postal.</li> <li>• <b>SUCC:</b> Succursale.</li> <li>• <b>PDF:</b> Poste de Facteurs.</li> <li>• <b>CPC:</b> Centre Postal Communautaire.</li> <li>• <b>CC:</b> Concession Commerciale.</li> </ul>
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").

Field	Description
Locality1 Locality2 Locality3 Locality4	The city, town or suburb and any additional related information.
Locality1_Official Locality2_Official Locality3_Official Locality4_Official	The locality name preferred by the postal authority.
Postcode	The postal code. <b>USA:</b> The five-digit ZIP Code and ZIP+4.
Postcode1	<b>Australia:</b> Four-digit postcode. <b>Canada:</b> First three characters (FSA) of the postal code. <b>Global:</b> Postal code. <b>USA:</b> Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4).
Postcode2	The secondary postal code. <b>Canada:</b> The last three characters (LDU) of the postal code. <b>USA:</b> 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.

Field	Description
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 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.  <b>E:</b> The record covers the even-numbered value.  <b>O:</b> The record covers the odd-numbered value.  <b>B:</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).

Field	Description
Region1	Returns the state, province, or region.
Secondary_Side_Indicator	<p>Indicates if even, odd, or both values are valid. This applies to floors and units.</p> <p><b>E:</b> The secondary record covers the even-numbered value.</p> <p><b>O:</b> The secondary record covers the odd-numbered value.</p> <p><b>B:</b> The secondary record covers both the even- and odd-numbered values.</p>
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").

### 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.

### 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's 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.
- This option does not support processing of Japanese or Chinese address data.

**Related Topics**

- [Global Suggestion List option groups](#)
- [Designer Guide: Cleanse your address data transactionally](#)

### 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 Topics**

- [Transform configurations](#)
- [Downloading blueprints and other content objects](#)

### 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 <b>Yes</b> . Select <b>No</b> to disable the country engine. Countries include: <ul style="list-style-type: none"> <li>Canada</li> <li>Global Address</li> <li>USA</li> </ul>
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 <a href="#">Options (Global Suggestion List)</a>
Suggestion List	See <a href="#">Suggestion List (Global Suggestion List)</a> .

### 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: <b>Yes:</b> Enables the engine. <b>No:</b> Disables the engine.
Global Address	Specifies if the engine is enabled or disabled for suggestion List processing. Choose one of the following: <b>Yes:</b> Enables the engine. <b>No:</b> Disables the engine.
USA	Specifies if the engine is enabled or disabled for suggestion List processing. Choose one of the following: <b>Yes:</b> Enables the engine. <b>No:</b> Disables the engine.

#### 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 <b>None</b> if you do not want to use a default country.

Option	Description
Enter Firm Data	<p>Specifies whether to request firm data when the selected suggestion does not have data available.</p> <p><b>Yes:</b> Requests firm data.</p> <p><b>No:</b> Does not request firm data.</p>
Return Single Item Lists	<p>Specifies whether the transform should return suggestion lists that have only one item.</p> <p><b>Yes:</b> Returns suggestion lists that only have one item.</p> <p><b>No:</b> If a suggestion list has only one item, then the single item is automatically selected from the suggestion list and processing continues.</p>

#### 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 5-121: Suggestion List options

Option/Option group	Description
Output Style	<p>Specifies the format for the output suggestion list data.</p> <p><b>Delimited:</b> Outputs the suggestion list data in a delimited text format, with the delimiters specified in the Delimiter and Field Delimiter options.</p> <p><b>XML:</b> 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.</p>

Option/Option group	Description
Delimiter	<p>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.</p> <p>This value can be any character or string. Common delimiters include a pipe symbol ( ), or a string of multiple asterisks (***).</p> <p>This value should differ from the Field Delimiter value.</p>
Field Delimiter	<p>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.</p> <p>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.</p> <p>This value can be any character or string. The default value is a pipe symbol ( ). This value should differ from the Delimiter value.</p>

Table 5-122: 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	<p>Indicates whether or not street data is available for a locality:</p> <p><b>Y:</b> Yes, there are streets.</p> <p><b>N:</b> No, there are not streets.</p>

Table 5-123: 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 5-124: 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. <b>E</b> : The record covers the even-numbered values. <b>O</b> : The record covers the odd-numbered values. <b>B</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.

### 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.
- 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.  <b>Important!</b> 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.



Field	Description
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.

### 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	<p>Returns a single-character code that indicates what type of additional data is needed to complete processing. Possible output values are:</p> <p><b>N:</b> No additional data is needed.</p> <p><b>A:</b> Primary address data is needed.</p> <p><b>F:</b> Firm data is needed.</p> <p><b>R:</b> Primary range data is needed.</p> <p><b>S:</b> Secondary range data is needed.</p>
Error	<p>Posts the error status generated as the result of looking up the current record and performing query processing. Possible output values are:</p> <p><b>0:</b> There were no query errors.</p> <p><b>1:</b> There was a system error while performing the query.</p> <p><b>2:</b> The suggestion selection was invalid. For example, a selection of 8 is made and there are only 5 entries.</p>

Field	Description
Firm	Specifies the firm name for the address.
Multiline1-6	<p>Returns a line that may contain any data. The type of data in this line may vary from record to record.</p> <p>If you want to output the postal code, you cannot use one of these Multiline output fields. You must use the Postcode1 field.</p>
Postcode1	<p>Returns the postcode.</p> <p><b>Canada and Global Address:</b> Postal code.</p> <p><b>USA:</b> Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4).</p>
Status	<p>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:</p> <p><b>C:</b> Querying is complete.</p> <p><b>D:</b> More data is needed.</p> <p><b>E:</b> There was an error.</p> <p><b>P:</b> A suggestion list was generated.</p>
Suggestion_Count	<p>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.</p> <p>Your application developer uses this field to know how many suggestion selections must be displayed to users of your custom application.</p>
Suggestion_List	Contains the list of suggestions based on the Suggestion List Option settings that you set in the Global Suggestion List transform.

Field	Description
Suggestion_Type	<p>Returns a code indicating what type of suggestion list was generated. Possible output values are:</p> <p><b>N:</b> No suggestion list was generated.</p> <p><b>A:</b> An address suggestion list was generated.</p> <p><b>L:</b> A lastline suggestion list was generated.</p> <p><b>S:</b> A secondary suggestion list was generated.</p>
System_Error_Description	Posts the current Global Suggestion List system error as a descriptive string.
System_Error_Number	<p>Posts the current system error as a number. The values are:</p> <p><b>0:</b> No error.</p> <p><b>1:</b> Invalid postcode.</p> <p><b>2:</b> Invalid street.</p> <p><b>3:</b> Invalid town.</p> <p><b>4:</b> More information needed.</p> <p><b>5:</b> Street information needed.</p> <p><b>6:</b> No input given.</p> <p><b>7:</b> Postcode numeric.</p> <p><b>8:</b> Town needed.</p> <p><b>9:</b> Town or postcode needed.</p> <p><b>10:</b> No street information available.</p> <p><b>11:</b> Country blank.</p> <p><b>12:</b> Invalid country.</p> <p><b>13:</b> No results.</p> <p><b>14:</b> Address needed.</p> <p><b>15:</b> Premise needed.</p> <p><b>16:</b> Firm needed.</p>

Field	Description
Warning	<p>Posts the warning status generated as the result of looking up the current record and performing query processing. The values are:</p> <p><b>0:</b> There were no query warnings.</p> <p><b>1:</b> An incomplete suggestion list was generated. Generally, the maximum number of elements that can be placed in a suggestion list is 200.</p> <p><b>2:</b> An invalid premise was entered.</p> <p><b>3:</b> An invalid unit was entered.</p>

### 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 Topics

- [Match transform tab](#)
- [Group forming](#)
- [Match level options](#)
- [Post-match processing](#)

### 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 Topics

- [Transform configurations](#)
- [Downloading blueprints and other content objects](#)

### 5.4.12.2 Match transform options

#### 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
<b>Perform matching</b>	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
<b>Match set name</b>	<p>Enter a name for this match set. Each Match transform in your data flow represents a match set.</p> <p>This option is already populated with name you chose if you used the Match wizard to generated this transform.</p> <p>Be sure that this name is unique within the dataflow (it does not match the name of another match set).</p> <p>This name is used in the match reports to differentiate data processed by one match set versus another.</p>
<b>Match engine</b>	<p>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.</p> <p><b>Chinese:</b> Specifies that the Match transform will be processing Chinese data in Chinese script.</p> <p><b>Japanese:</b> Specifies that the Match transform will be processing Japanese data in Japanese script.</p> <p><b>Korean:</b> Specifies that the Match transform will be processing Korean data in Korean script.</p> <p><b>Latin1:</b> 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.</p> <p><b>Other_Non_Latin1:</b> 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.</p> <p><b>Taiwanese:</b> Specifies that the Match transform will be processing Taiwanese data in Taiwanese script.</p> <p>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.</p> <p>By default, the Latin1 match engine is used. If you attempt to process non-latin1 data with the Latin1 engine, the results are unpredictable.</p>

Option	Description
<b>Generate report data</b>	<p>Specifies whether to generate report data for this transform. This option is available in every transform that generates report data.</p> <p><b>Yes:</b> Generates report data for this transform.</p> <p><b>No:</b> Turns off report data generation. If you do not need to generate reports (during testing of dataflows, for example), you should set this option to No. This will improve match performance.</p>
<b>Logical source field</b>	Specifies the field that contains the ID for the logical source.
<b>Physical source field</b>	Specifies the field that contains the ID for the physical source (Reader).
<b>Run as a separate process</b>	<p><b>Yes:</b> Splits the transform into a separate process.</p> <p><b>No:</b> Keeps the transform in same process as the rest of the dataflow.</p>

### Related Topics

- [Performance Optimization Guide: Run as a separate process option](#)

#### 5.4.12.2.2 Match transform options: Data Salvage tab

Option	Description
<b>Match set name</b>	<p>Enter a name for this match set. Each Match transform in your data flow represents a match set.</p> <p>This option is already populated with name you chose if you used the Match wizard to generated this transform.</p> <p>Be sure that this name is unique within the dataflow (it does not match the name of another match set).</p> <p>This name is used in the match reports to differentiate data processed by one match set versus another.</p>
<b>Enable data salvage</b>	<p>Select to perform data salvaging.</p> <p>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.</p>

Option	Description
<b>Perform data salvage default</b>	<p>Specifies the default value that indicates whether to perform data salvage if the data record does not contain a field with this value.</p> <p><b>Yes:</b> Performs the data salvage on the driver record after it matches a passenger record.</p> <p><b>No:</b> Does not perform the data salvage on the driver record after it matches a passenger record.</p>
<b>Specify data salvage by field</b>	Select to control data salvaging by means of a value in a field.
<b>Perform data salvage field</b>	Specifies the field that contains the indicator for performing the data salvage operation. Use this to override the default value.
<b>Specify data salvage by source</b>	Select to control data salvaging per source.
<b>Source</b>	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.
<b>Perform data salvage</b>	Select Yes or No to signify you want data salvaging performed on a source.

### Related Topics

- [Designer Guide: Match, Data Salvage](#)

#### 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
<b>Value field</b>	Choose an input field from the drop-down list that contains the value for your sources.
<b>Source name</b>	Enter a name for your input source.
<b>Source value</b>	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.



Option	Description
<b>Source type</b>	<p>Choose a source type from the drop-down list.</p> <p><b>Normal:</b> A Normal source contains good or eligible records.</p> <p><b>Suppress:</b> 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:</p> <ul style="list-style-type: none"><li>• DMA Mail Preference File</li><li>• American Correctional Association prisons/jails lists</li><li>• No pandering or non-responder lists</li><li>• Credit card or bad-check suppression lists</li></ul> <p><b>Special:</b> 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.</p> <p>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.</p>
<b>Default source name</b>	<p>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.</p> <p>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).</p>

Option	Description
<b>Auto generate sources</b>	<p>Select to create sources for each unique entry in the Value field.</p> <p>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.</p> <p>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.</p> <p><b>Note:</b> 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.</p>
<b>Default type</b>	<p>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.</p>
<b>Type field</b>	<p>Choose an input field from the drop-down list that contains the input source type.</p> <p>The type field must have a value of:</p> <p><b>N:</b> Normal</p> <p><b>P:</b> Suppress (or Purge)</p> <p><b>S:</b> Special</p> <p>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.</p> <p><b>Note:</b> This option is not case sensitive.</p>

#### 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
<b>Source groups</b>	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.
<b>Sources</b>	This list contains all of the sources defined in the Input Sources operation.
<b>Undefined action</b>	<p>Specifies the action to take if an input source does not appear in a source group.</p> <p><b>Ignore:</b> The input source does not belong to any source group.</p> <p><b>Default:</b> The input source belongs to the default source group specified in the <b>Default source group</b> option.</p> <p><b>Auto:</b></p> <ul style="list-style-type: none"> <li>• If the <b>Source group field</b> 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.</li> <li>• If the <b>Source group field</b> option is defined, then the input source belongs to the source group named in the <b>Source group field</b> option. The source group is created if necessary.</li> <li>• If the source group field's content is blank, then that input source will not belong to a source group (equivalent of Ignore).</li> </ul>
<b>Default source group</b>	<p>Select a default source group name from the drop-down list. You can choose from your defined source groups.</p> <p>This option is required if you chose Default as the value for the <b>Undefined action</b> option.</p>
<b>Source group field</b>	Specifies the field that contains the value for your source groups.

### 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.

#### 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 Topics

- [Break group options](#)
- [Candidate selection options](#)
- [Group prioritization options: Priority Order tab](#)

#### 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	<p>Select this option if you want to form break groups to reduce the total number of comparisons made.</p> <p>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.</p> <p><b>Caution:</b> 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 <b>Maximum allowable break group size (in records)</b> option so that the collection does not exceed the size limit. If it does exceed the limit, the data flow will abort.</p> <p><b>Note:</b> Break group size is calculated by multiplying the record length by the number of records in the break group.</p>

Options	Description
<b>Field</b>	<p>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.</p> <p>If you require a more complex break key, you could define that field using an upstream Query transform and select the field here.</p>
<b>Start Position</b>	<p>Enter the start position of the field. Valid values for a field of <math>n</math> are 1 to <math>n</math> and -1 to <math>-n</math>. Negative start values signify that the start position is counted from the right.</p> <p>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).</p>
<b>Length</b>	Enter the number of characters in the field you want included in the break key.
<b>Break key case sensitive</b>	<p>Specifies whether to treat the break key as case sensitive.</p> <p><b>Yes:</b> Treat the break key as case sensitive.</p> <p><b>No:</b> Do not treat the break key as case sensitive.</p> <p>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.</p>
<b>Replace NULL with empty string</b>	<p>Specifies whether to convert NULL values with an empty string in the break key.</p> <p><b>Yes:</b> Convert NULL to an empty string.</p> <p><b>No:</b> Do not convert to an empty string.</p>
<b>Right pad fields with blanks</b>	<p>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.</p> <p>If the <b>Replace NULL with empty string</b> 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).</p> <p><b>Yes:</b> Right-pad fields with blank spaces.</p> <p><b>No:</b> Do not right-pad fields.</p>

Options	Description
<b>Input already sorted</b>	<p>Specifies that the input data has already been sorted, and you do not want it sorted again.</p> <p>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.</p> <p><b>Yes:</b> The transform will not re-sort the input data.</p> <p><b>No:</b> The transform will sort the break keys at runtime before forming break groups.</p>
<b>Maximum allowable break group size (in records)</b>	<p>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.</p> <p>With this option, you can control the amount of memory used during processing by specifying the number of records processed at one time.</p> <p>If more records make it into a single break group than specified, then the dataflow throws an error and stops.</p>

### Related Topics

- [Designer Guide: Match, Break keys and candidate selection](#)

#### 5.4.12.3.2 Candidate selection options

The candidate selection option group includes the following options:

Option	Description
<b>Datastore</b>	<p>Select a valid datastore.</p> <p>This list is populated with all valid SQL and persistent cache datastores.</p> <p>If you choose a persistent cache datastore, you will not be able to enter custom SQL.</p>
<b>Cache type</b>	<p>This option can be used to improve performance, with a trade-off of more memory consumption.</p> <p><b>No_Cache:</b> Specifies that each query will be sent to the database.</p> <p><b>Pre_Load_Cache:</b> Specifies that the entire secondary table is cached to a local disk or memory.</p>

Option	Description
<b>Auto-generate SQL</b>	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 <b>Create custom SQL</b> option.
<b>Table</b>	Enter a valid table name from the datastore.
<b>Use break column from database</b>	Select this option if your database already contains a column that corresponds to the break key field.
<b>Break key field</b>	Select the column from the secondary table that contains the break key field.
<b>Create custom SQL</b>	Select to create custom SQL.
<b>Launch SQL Editor</b>	Opens the SQL editor. This button is only enabled if you select the <b>Create custom SQL</b> option.
<b>Use constant source value</b>	Select to assign records to a physical source for generating appropriate statistics.
<b>Physical source value</b>	Type a value for your physical source. This value will be placed in the physical source field you select.
<b>Physical source field</b>	Select the mapped field that contains the physical source name.
<b>Add DB columns to mapping table</b>	<p>If you are using the <b>Create custom SQL</b> 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.</p> <p>If you are using the <b>Auto-generate SQL</b> option, clicking this button will add ALL database columns, in the order that they appear in the table schema.</p> <p><b>Note:</b></p> <p>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.</p>

### Column mapping table

This table allows you to specify which mapped field in the dataflow each database selected field is assigned to.

Column	Description
<b>Break key</b>	Specifies whether this field is used as part of your break key.

Column	Description
<b>Field</b>	Each cell contains a list of the mapped names from the input fields in the transform.
<b>DB column</b>	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 Topics

- [Designer Guide: Match, Break keys and candidate selection](#)

#### 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
<b>Prioritization name</b>	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
<b>Input field</b>	Choose a field to sort your records on.
<b>Field order</b>	Specifies in which order records should be sorted.



## 5.4.12.3.4 Group prioritization options: Record Completeness tab

Option	Description
<b>Prioritization name</b>	Specifies the name for this Group prioritization operation. If you have multiple operations in this Match transform, be sure to make this name unique.
<b>Order records based on completeness of data</b>	Select this option to apply priority and blank penalty points to records to help control the order of your records.
<b>Define only field penalties</b>	Select this option when penalties need to be assessed based on blank fields.
<b>Define priority and penalty fields</b>	Select this option when you have specific fields that contain the actual integer values for priority and blank penalty.
<b>Record priority field</b>	Choose the field that contains priority values. This field must contain an integer.
<b>Apply blank penalty field</b>	Choose the field that contains the indicator (Y or N) for applying blank penalty points to a record.
<b>Define priority and penalty based on input source</b>	Select to have your record priority and blank penalty indicator (Y or N) determined by membership in a given source.
<b>Source Name</b>	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.
<b>Priority</b>	Type a priority value (an integer) in the Priority column. Remember that the lower the priority score, the higher the priority.
<b>Apply Blank Penalty</b>	Choose Yes or No to determine whether a blank penalty is applied to a record based on membership to this source.
<b>Default record priority</b>	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
<b>Default apply blank penalty</b>	<p>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 if a record does not belong to any of the sources specified.</p> <p><b>Yes:</b> 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.</p> <p><b>No:</b> No penalty is applied when the fields are blank.</p>
<b>Input field</b>	Displays the input fields available to assign a blank penalty score to.
<b>Blank penalty</b>	Assign a penalty value (an integer) to apply when the specified field is blank in a record.

## 5.4.12.4 Matching

### 5.4.12.4.1 Match level options

These options affect processing at the match level only.

Table 5-137: Person Options

Option	Description
<b>Match level name</b>	Enter a name for this match level. Be sure that this name is unique within the Match transform.
<b>Weighted match score</b>	<p>Specifies the weighted match score for this level.</p> <p>When your matching method includes weighted scoring, records are considered matches when the total contribution score is greater than or equal to this value.</p>

Option	Description						
<b>Number of names that must match</b>	<p>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.</p> <p><b>One:</b> Specifies that records are a match when at least one of the names meet the criteria.</p> <p><b>All:</b> Specifies that records are a match only when all of the names meet the criteria.</p>						
<b>Match on hyphenated family name</b>	<p>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.</p> <p>This comparison is performed only if one field has a hyphen and the other does not.</p> <table data-bbox="528 825 1437 972"> <tr> <th data-bbox="528 825 981 871">Given</th><th data-bbox="981 825 1437 871">Family</th></tr> <tr> <td data-bbox="528 871 981 919">Laura</td><td data-bbox="981 871 1437 919">Smith</td></tr> <tr> <td data-bbox="528 919 981 972">Laura</td><td data-bbox="981 919 1437 972">Albers-Smith</td></tr> </table> <p>This option works on a criteria named Family_Name1, Family_Name2, or Family_Name3.</p> <p><b>Yes:</b> The family names match as long as the single family name in one record matches one of the hyphenated family names in another record.</p> <p><b>No:</b> 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.</p>	Given	Family	Laura	Smith	Laura	Albers-Smith
Given	Family						
Laura	Smith						
Laura	Albers-Smith						

Option	Description									
Compare Given Name1 to Given Name2	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.									
	<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>	Given name1	Given name2	Family name1	John		Smith	R	John	Smith
	Given name1	Given name2	Family name1							
	John		Smith							
R	John	Smith								
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.										
<p><b>Yes:</b> The Given_Name1 field of one record is compared to the Given_Name2 field of another record.</p> <p><b>No:</b> The Given_Name1 field of one record is not compared to the Given_Name2 field of another record.</p>										
Ignore family name when female	<p>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.</p> <p><b>Yes:</b> The Family_Name1 criteria is ignored when the given name gender is a female (Gender=5). For example, Laura Smith may match Laura Albers.</p> <p><b>No:</b> The gender is not used and the matching process is performed as usual.</p>									

Table 5-138: Address Options

Option	Description																		
Match on Street and RR, or on Box	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.																		
	<b>Yes:</b> 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 Hard-ware</td><td>100</td><td>Elm</td><td>Ave</td><td>200</td><td>02961</td></tr><tr><td>Acme Hard-ware</td><td>123</td><td>Main</td><td>St</td><td>200</td><td>02961</td></tr></table>	firm	number	street	suffix	postal number	postcode	Acme Hard-ware	100	Elm	Ave	200	02961	Acme Hard-ware	123	Main	St	200	02961
	firm	number	street	suffix	postal number	postcode													
Acme Hard-ware	100	Elm	Ave	200	02961														
Acme Hard-ware	123	Main	St	200	02961														
<b>No:</b> All forms of the address (street, rural route, and PO Box) must match.																			

Option	Description																		
Address matches blank if Firms match	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.																		
	<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 Hard-ware</td><td>100</td><td>Elm</td><td>Ave</td><td></td><td>02961</td></tr><tr><td>Acme Hard-ware</td><td></td><td></td><td></td><td>300</td><td></td></tr></table>	firm	number	street	suffix	postal number	postcode	Acme Hard-ware	100	Elm	Ave		02961	Acme Hard-ware				300	
	firm	number	street	suffix	postal number	postcode													
	Acme Hard-ware	100	Elm	Ave		02961													
Acme Hard-ware				300															
<p><b>Yes:</b> If firm data matches and neither firm field is blank, blank matching is allowed for all address components.</p> <p><b>No:</b> 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>																			
Unique on resident if RR, but no Box	<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><b>Yes:</b> Places all records with this type of name data into the same match group.</p> <p><b>No:</b> Does not place records with this type of name data into the same match group.</p>																		

Option	Description
<b>Ignore Firm if Name matches</b>	<p>This option works with odd abbreviations or spellings of firm names. This assumes that you are matching on two Family Names.</p> <p><b>Yes:</b> 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><b>No:</b> Both the firm criteria and the address criteria must meet the minimum similarity threshold in order to match.</p>

**Related Topics**

- [Designer Guide: Match, Weighted scoring method](#)

**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.

**Related Topics**

- [Match criteria options: Criteria Fields tab](#)

**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	<p><b>Address_Data1-5:</b> Use for address data that is not accounted for in other address-based criteria in the Geographic category.</p> <p>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.</p>
	<b>Address_Post_Office_Box:</b> Post Office box number.
	<b>Address_Primary_Name:</b> Street name data.
	<b>Address_Primary_Number:</b> Street number data.
	<b>Address_Primary_Postfix:</b> Address data that comes at the end of a street name, such as a directional.
	<b>Address_Primary_Prefix:</b> Address data that comes at the beginning of a street name, such as a directional.
	<b>Address_Primary_Type:</b> Data that tells what type of street it is (street, boulevard, lane, and so on).
	<b>Address_Private_Mail_Box:</b> A private mail box (PMB) number. These are mail boxes that are not run by a postal authority.
	<b>Address_Rural_Route_Box:</b> Rural-route box number (number only, without "Box" prefix).
	<b>Address_Rural_Route_Number:</b> Rural route number.
	<b>Address_Secondary_Number:</b> The number of a unit, building, floor, or room.
	<b>Country:</b> Country name.
	<b>Locality:</b> City, town, locality, or suburb.
	<b>Latitude_Longitude:</b> Latitude and longitude.



Category	Description
	<b>Postcode1:</b> Primary postal code.
	<b>Postcode2:</b> Secondary postal code.
	<b>Region:</b> Region data, such as state or province.
<b>Firm</b>	<b>Firm:</b> Firm name.
	<b>Firm_Data1-3:</b> 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.
	<b>Firm_Match_Std1-6:</b> Firm match standards. The data in these fields is generated by the Data Cleanse transform or other pre-Match transforms.
	<b>Firm_Location:</b> A location within a company or organization.
	<b>Firm_Location_Match_Std1-6</b> Match standards for a location within a company or organization.

Category	Description
Person	<b>Name_Data1-3:</b> 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.
	<b>Person1-3_Given_Name1:</b> The given name1 (first name) of the persons.
	<b>Person1_Given_Name1_Match_Std1-6:</b> Given_Name1 (first name) match standards for the first person.
	<b>Person2_Given_Name1_Match_Std1-6:</b> Given_Name1 (first name) match standards for the second person.
	<b>Person3_Given_Name1_Match_Std1-6:</b> Given _Name1 (first name) match standards for the third person.
	<b>Person1-3_Gender:</b> Gender.
	<b>Person1-3_Family_Name1:</b> Family (last) name.
	<b>Person1-3_Family_Name1_Match_Std1-6:</b> Family_Name1 match standards for the persons in your data record.
	<b>Person1-3_Family_Name2:</b> 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.
	<b>Person1-3_Family_Name2_Match_Std1-6:</b> Family_Name2 match standards for the persons in your data record.
	<b>Person1-3_Maturity_Postname:</b> Maturity postname.. For example, Sr. or Jr. (one standard per person).
	<b>Person1-3_Maturity_Postname_Match_Std1-6:</b> Maturity postname match standards for the persons in your data record.
	<b>Person1-3_Given_Name2:</b> Given name2 (middle name).

Category	Description
	<b>Person1_Given_Name2_Match_Std1-6:</b> Given_Name2 (middle name) match standards for the first person.
	<b>Person2_Given_Name2_Match_Std1-6:</b> Given_Name2 (middle name) match standards for the second person.
	<b>Person3_Given_Name2_Match_Std1-6:</b> Given_Name2 (middle name) match standards for the third person.
	<b>Person1-3_Honorary_Postname:</b> Honorary postname for up to three persons indicating certification, academic degree, or affiliation. For example, CPA.
	<b>Person1-3_Honorary_Postname_Match_Std1-6:</b> Honorary postname match standards.
	<b>Person1-3_Prenome:</b> Prenome (for example, Mr. or Mrs.) for up to three persons.
	<b>Person1-3_Prenome_Match_Std1-6:</b> Prenome match standards.
	<b>Person1-3_Title:</b> Job or occupational title of each person. For example, Manager.
	<b>Person1-3_Title_Match_Std1-6:</b> Title match standards for each person.
	<b>Social_Security_Number1-3:</b> Social Security numbers for up to three people in a record.
<b>Other</b>	<b>Date1-3:</b> Date data. For example, birthdate data.
	<b>Phone:</b> Phone number.
<b>Custom</b>	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
<b>Criteria field</b>	The criteria field that contains the data you want to compare.
<b>Input field mapped name</b>	<p>Choose the mapped name for your criteria field.</p> <p>If you choose any of the following input fields, the Match transform automatically adds the appropriate match standard fields.</p> <p>The fields displayed will vary depending on the value chosen for <b>Compare data using</b> on the Options tab.</p> <ul style="list-style-type: none"> <li>• Firm</li> <li>• Firm_Location</li> <li>• Person*_Given_Name1</li> <li>• Person*_Given_Name2</li> <li>• Person*_Honorary_Postname</li> <li>• Person*_Maturity_Postname</li> <li>• Person*_Prenome</li> <li>• Person*_Title</li> </ul> <p>You can then choose to include or exclude any of these from mapping.</p> <p><b>Note:</b></p> <p>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 <b>Additional fields to match</b> table in the "Multiple field matching" section.</p>

*Match criteria options: Options tab*

Option	Description
<b>Compare data using</b>	<p>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.</p> <p><b>Field Similarity:</b> 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.</p> <p><b>Geo Proximity:</b> 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.</p> <p><b>Numeric Difference:</b> 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.</p> <p><b>Numeric Percent Difference:</b> 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.</p> <p><b>Word Similarity:</b> 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.</p> <p>Many criteria options require this option to be set to Word Similarity.</p>

**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
<b>Field compare length</b>	<p>Specifies the number of characters in the field to compare.</p> <p><b>Note:</b> 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.</p>

Option	Description
<b>Remove punctuation</b>	<p>Specifies whether to remove punctuation from your data to help provide more accurate matches. Be aware of the following:</p> <ul style="list-style-type: none"> <li>• This option is valid for Latin1 data only.</li> <li>• Match will not remove a dash from a Family_Name* field.</li> </ul> <p><b>Yes:</b> Removes punctuation.</p> <p><b>No:</b> Keeps punctuation in your data.</p> <p><b>Note:</b> 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.</p> <p><b>Caution:</b> Setting this option and the <b>Convert text to numbers</b> option to <b>Yes</b> may produce undesirable results. For example:</p> <p>Suppose you have 1.23 as data in a criteria field. Setting <b>Remove punctuation</b> to <b>Yes</b> 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".</p>
<b>Convert to uppercase</b>	<p>Specifies whether to convert all data to uppercase for matching purposes only. This option is valid for Latin1 data only.</p> <p><b>Yes:</b> Converts the data to uppercase for the comparison process.</p> <p><b>No:</b> Preserves the case of the data for the comparison process, treating A and a as different characters.</p> <p><b>Note:</b> 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.</p>

Option	Description
<b>Convert diacritical characters</b>	<p>Specifies whether to include diacritical characters in the matching process. Be aware of the following:</p> <ul style="list-style-type: none"> <li>• This option is valid for all match engine options.</li> <li>• 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.</li> </ul> <p><b>Yes:</b> Converts diacritical characters to the closest English ASCII equivalent for matching purposes. For example, ä converts to a.</p> <p><b>No:</b> Preserves diacritical characters in the matching process, treating ä and a as not identical characters.</p> <p><b>Note:</b> 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.</p>
<b>Convert text to numbers</b>	<p>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.</p> <p><b>Yes:</b> Converts numbers represented as text to numbers.</p> <p><b>No:</b> Leaves any numerical text intact.</p> <p><b>Note:</b> 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.</p> <p><b>Caution:</b> Setting this option and the <b>Remove punctuation</b> option to <b>Yes</b> may produce undesirable results. For example:</p> <p>Suppose you have 1.23 as data in a criteria field. Setting <b>Remove punctuation</b> to <b>Yes</b> 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".</p>

Option	Description
<b>Locale</b>	<p>Specifies the locale setting for this criteria field. If nothing is specified, or you specify <b>DEFAULT</b>, the default system locale in the software will be used.</p> <p>Setting this option is recommended if you plan to use the <b>Convert text to numbers</b> option.</p>

### Comparison rules

Option	Description						
<b>Approx substring adjustment score</b>	<p>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 <b>Substring adjustment score</b> option is too strict to compare, such as "Cruz Rodríguez" and "C. de Rodríguez".</p> <p>Enter a value from 0 (default) to 100. Enter a value of 0 to disable the option.</p>						
<b>Abbreviation adjustment score</b>	<p>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.</p> <p>For example:</p> <table border="1"> <thead> <tr> <th>Full word</th><th>Possible abbreviations</th></tr> </thead> <tbody> <tr> <td>Business</td><td>Bus, Bsnss, Bss</td></tr> <tr> <td>Database</td><td>Dat, Db, Dse</td></tr> </tbody> </table> <p>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.</p> <ul style="list-style-type: none"> <li>• Enter a value of 0 (zero) to disable abbreviation checking.</li> <li>• Enter a value greater than 0 to enable this option.</li> <li>• Enter a value of 100 if you want abbreviations and longer words to be considered a perfect match.</li> </ul> <p><b>Note:</b> For this option to work, you must set the <b>Compare data using</b> option to <b>Word Similarity</b>.</p>	Full word	Possible abbreviations	Business	Bus, Bsnss, Bss	Database	Dat, Db, Dse
Full word	Possible abbreviations						
Business	Bus, Bsnss, Bss						
Database	Dat, Db, Dse						



Option	Description
<b>Both fields blank operation</b>	<p>Specifies whether to use this criteria when both of the records' fields for this criteria are blank.</p> <p><b>Eval:</b> The value entered in the Both fields blank score option is used as the similarity score for this criteria.</p> <p><b>Ignore:</b> 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.</p>
<b>Both fields blank score</b>	<p>Specifies the similarity score if both of the fields are blank and Both fields blank operation is set to <b>Eval</b>.</p> <p>Enter a value from 0 to 100.</p>
<b>Check for transposed letters</b>	<p>Specifies whether the match score should be adjusted for any transposed characters encountered.</p> <p><b>Yes:</b> The transform deducts half as many points for transposed characters as it deducts for other non-matching characters.</p> <p>For example:</p> <p>Comparison: Smith—Simth</p> <p>Finding: Words differ by one transposition (penalty of 1 correction)</p> <p>Percentage alike: 90%</p> <p><b>No:</b> The transform handles transposed characters the same way it handles any non-matching characters.</p> <p>For example:</p> <p>Comparison: Smith—Simth</p> <p>Finding: Words differ by one transposition (penalty of 1 correction)</p> <p>Percentage alike: 80%</p>

Option	Description
<b>Contribution to weighted score</b>	<p>Specifies the contribution value, when you use the weighted or the combination scoring method.</p> <p>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.</p> <p>Type a value between 0 and 100.</p>
<b>Distance Unit</b>	<p>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:</p> <ul style="list-style-type: none"> <li>• <b>Feet</b></li> <li>• <b>Kilometers</b></li> <li>• <b>Meters</b></li> <li>• <b>Miles</b></li> </ul>
<b>Enable inter-script matching</b>	<p>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 .</p>
<b>Ext abbreviation adjustment score</b>	<p>This option handles a variation of the <b>Abbreviation adjustment score</b> option.</p> <p>Enter a number that adjusts the similarity score for these types of abbreviations. For example:</p> <ul style="list-style-type: none"> <li>• Enter a value of 0 (default) to disable abbreviation checking.</li> <li>• Enter a value greater than 0 to enable abbreviation checking.</li> </ul> <p>Remember the following when using this option:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• Letters that match are given a score of 100. The remaining letters are given the score that you specify.</li> <li>• The two scores are proportionally combined to render the overall score.</li> </ul> <p><b>Note:</b> For this option to work, you must set the <b>Compare data using</b> option to <b>Word Similarity</b>.</p>

Option	Description
<b>Initials adjustment score</b>	<p>Specifies whether you want initials or acronyms to match whole words. For example, the firm name International Health Providers could match IHP.</p> <p>Enter a value from 0 to 100. Enter a value of 0 (default) to disable initial checking.</p> <p>Remember the following when using this option:</p> <ul style="list-style-type: none"> <li>• The initial must match the first letter of the word.</li> <li>• The letters that match are given a score of 100. The remaining letters are given the score that you specify (from 1-100).</li> <li>• The two scores are proportionally combined to render the overall score.</li> </ul> <p>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.</p> <p><b>Note:</b> For this option to work for multiple-word abbreviations (such as International Health Providers = IHP) you must set the <b>Compare data using</b> option to Word Similarity. For this option to work for single-word abbreviations (such as Maria = M) you may set the <b>Compare data using</b> option to either Word Similarity or Field Similarity..</p>
<b>Match score</b>	<p>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.</p> <p>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.</p> <p>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.</p>
<b>No match score</b>	<p>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.</p> <p>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.</p> <p>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.</p>

Option	Description
<b>Max Difference</b>	Specifies the maximum difference allowed in a numeric range. Type a value from 0 - 2147483647.
<b>Max Difference Score</b>	<p>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.</p> <p>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.</p> <p>Any difference less than Max Difference will receive a proportional score between Max Difference Score and 100.</p>
<b>Max Distance</b>	Specifies the maximum distance allowed when calculating the distance between two Latitude, Longitude pairs. Type a value from 0 to 4294967295.0.
<b>Max Distance Score</b>	<p>Specifies what score to generate when the distance is the same as Max Distance. Type a value from 0 to 100.</p> <p>Any distance larger than the Max Distance will receive a score of 0.</p> <p>A distance equal to Max Distance will receive a score of Max Distance Score.</p> <p>Any distance less than Max Distance will receive a proportional score between Max Distance Score and 100.</p>
<b>Max Percent Difference</b>	Specifies the maximum difference allowed as a percent of the absolute value. Type a value from 0 to 100.
<b>Max Percent Difference Score</b>	<p>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.</p> <p>Any difference larger than the Max Percent Difference will receive a score of 0.</p> <p>A difference equal to Max Percent Difference will receive a score of Max Percent Difference Score.</p> <p>Any difference less than Max Percent Difference will receive a proportional score between Max Percent Difference Score and 100.</p>

Option	Description
<b>Numeric words match exactly</b>	<p>Specifies how to match data that contains both numbers and letters.</p> <p><b>None:</b> Numeric words don't need to match exactly to be considered a match.</p> <p><b>Any_Position:</b> Numeric words don't need to be in the same position in two different strings to be considered a match.</p> <p><b>Any_Position_Consider_Punctuation:</b> 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.</p> <p><b>Any_Position_Ignore_Punctuation:</b> Same as Any_Position, except that decimal separators (comma or period) are completely ignored.</p> <p><b>Same_Position:</b> Numeric words must match exactly and be in the same position in the string to be considered a match.</p> <p><b>Note:</b> For this option to work, you must set the <b>Compare data using</b> option to <b>Word Similarity</b>.</p>
<b>One field blank operation</b>	<p>Specifies whether to use this criteria if one record's field is populated and the other record's field is blank.</p> <p><b>Eval:</b> The value entered in the One field blank score is used as the similarity score for this criteria.</p> <p><b>Ignore:</b> This criteria is ignored in the comparison process, and its contribution to the weighted score is proportionally distributed among the remaining criteria.</p>
<b>One field blank score</b>	<p>Specifies the similarity score to use if one of the fields is blank and the <b>One field blank operation</b> option is set to <b>Eval</b>.</p> <p>Type a value from 0 to 100.</p>

Option	Description
<b>Substring adjustment score</b>	<p>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.</p> <p>Remember the following rules about values to enter:</p> <ul style="list-style-type: none"> <li>• Enter a value from 0 to 100. Enter a value of 0 (default) to disable substring checking.</li> <li>• Enter a value of 100 if you want substrings and longer strings to be considered a match.</li> </ul> <p>Here is what happens after processing.</p> <ul style="list-style-type: none"> <li>• Letters that match are given a score of 100. The remaining letters are given the score you specify (from 1-100).</li> <li>• The two scores are proportionally combined to render the overall score.</li> </ul> <p>To qualify as a substring match, the shorter string must exactly match the first part of the longer string.</p> <p>Consider the following example:</p> <p>Matching substrings</p> <ul style="list-style-type: none"> <li>• Mayfield</li> <li>• Mayfield Painting</li> <li>• Mayfield Painting and Sand</li> </ul> <p>Substrings that do not match</p> <ul style="list-style-type: none"> <li>• Mayfield Sand Blasting</li> <li>• Painting and Sand Blasting</li> </ul> <p>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.)</p> <p><b>Note:</b> For this option to work, you must set the <b>Compare data using</b> option to <b>Word Similarity</b>.</p>
<b>Use in weighted score if greater than</b>	<p>Specifies the minimum similarity score needed to qualify this criteria to contribute to the Weighted Match Score.</p> <p>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.</p>

Option	Description
<b>Zero weighted score if less or equal</b>	<p>Specifies the minimum similarity score needed for this criteria to qualify for contributing a value other than zero to the weighted match score.</p> <p>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.</p>

### 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 Topics

- [Designer Guide: Match, Numeric data matching](#)
- [Designer Guide: Match, Extended abbreviation matching](#)
- [Designer Guide: Match, Matching methods](#)
- [Designer Guide: Match, Unicode matching](#)

*Match criteria options: Multiple Field Comparisons tab***Multiple field mapping**

Option	Description
<b>Compare multiple fields</b>	Enable multiple field matching.
<b>All selected fields in other records</b>	Match each field against all fields selected in the table in each record.
<b>The same field in other records</b>	Match each field only against the same field in each record.

**Additional fields to compare**

Option	Description
<b>Criteria field</b>	Choose the input field that contains the data you want to compare.
<b>Custom name</b>	Enter a name for your custom field.
<b>Input field mapped name</b>	Choose the mapped input field name for the criteria field. The fields displayed will vary depending on the value chosen for <b>Compare data using</b> 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
<b>Field compare length</b>	Specifies the number of characters in the field to compare.



Option	Description
<b>Remove punctuation</b>	<p>Specifies whether to remove punctuation from your data to help provide more accurate matches. Be aware of the following:</p> <ul style="list-style-type: none"> <li>• This option is valid for Latin1 data only.</li> <li>• Match will not remove a dash from a Family_Name* field.</li> </ul> <p><b>Yes:</b> Removes punctuation.</p> <p><b>No:</b> Keeps punctuation in your data.</p> <p><b>Note:</b> 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.</p> <p><b>Caution:</b> Setting this option and the <b>Convert text to numbers</b> option to <b>Yes</b> may produce undesirable results. For example:</p> <p>Suppose you have 1.23 as data in a criteria field. Setting <b>Remove punctuation</b> to <b>Yes</b> 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".</p>
<b>Convert to uppercase</b>	<p>Specifies whether to convert all data to uppercase for matching purposes only. Be aware of the following:</p> <ul style="list-style-type: none"> <li>• This option is valid for English language, Latin1 data only.</li> <li>• This option is ignored for all other Match engine option values.</li> </ul> <p><b>Yes:</b> Converts the output data to uppercase where appropriate.</p> <p><b>No:</b> Leaves the output data intact.</p> <p><b>Note:</b> 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.</p>

Option	Description
<b>Convert diacritical characters</b>	<p>Specifies whether to include diacritical characters in the matching process. Be aware of the following:</p> <ul style="list-style-type: none"> <li>• This option is valid for all match engine options.</li> <li>• 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.</li> </ul> <p><b>Yes:</b> Converts diacritical characters to the closest English ASCII equivalent for matching purposes. For example, ä converts to a.</p> <p><b>No:</b> Preserves diacritical characters in the matching process, treating ä and a as not identical characters.</p> <p><b>Note:</b> 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.</p>
<b>Convert text to numbers</b>	<p>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.</p> <p><b>Yes:</b> Converts numbers represented as text to numbers.</p> <p><b>No:</b> Leaves any numerical text intact.</p> <p><b>Note:</b> 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.</p> <p><b>Caution:</b> Setting this option and the <b>Remove punctuation</b> option to <b>Yes</b> may produce undesirable results. For example:</p> <p>Suppose you have 1.23 as data in a criteria field. Setting <b>Remove punctuation</b> to <b>Yes</b> 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".</p>
<b>Locale</b>	<p>Specifies the locale setting for this criteria field. If nothing is specified, or you specify <b>DEFAULT</b>, the default system locale in SAP BusinessObjects Data Services will be used.</p> <p>Setting this option is recommended if you plan to use the <b>Convert text to numbers</b> option.</p>

#### 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
<b>Default action</b>	<p>Specifies the action assigned to each cell of the compare table initially before any Compare actions table rows are applied.</p> <p><b>Compare:</b> 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.</p> <p><b>No_Match:</b> Does not compare all logical sources in the compare table. You can then specify a pair to be compared in the Compare actions table.</p>
<b>Default source</b>	<p>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.</p>
<b>Define compare actions using field values</b>	<p>Select to use values in a field as the criteria for comparison, rather than membership in an input source. To use this option, you will need to enter field values in the <b>Compare Actions</b> table.</p> <p>This option is automatically selected if you do not have any input sources defined.</p>
<b>Logical source field</b>	<p>Specifies the field that contains the logical source value (name).</p>
<b>Default logical source value</b>	<p>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.</p> <p>This option specifies which value defined in the Compare actions table is the default value.</p>

### 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
<b>Driver source</b>	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.
<b>Passenger source</b>	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.
<b>Action</b>	<p>Specifies the action to take when a record from the Driver source is to be compared with a record from the Passenger source.</p> <p><b>Compare:</b> Compare the two records</p> <p><b>No_Match:</b> Do not compare the two records.</p> <p>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.</p>

#### 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 Topics

- [Best record options: Best Record tab](#)
- [Group statistics options](#)
- [Group prioritization options: Priority Order tab](#)
- [Unique ID options: Unique ID tab](#)
- [Output flag selection options](#)
- [Designer Guide: Match, Best record](#)
- [Designer Guide: Match, Unique ID](#)

### *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
<b>Best record name</b>	Enter a name for your Best Record operation. Make sure that this name is unique within this Match transform.

Option	Description
<b>Best record strategy</b>	<p>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).</p> <p><b>Custom:</b> 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.</p> <p><b>Date:</b> Choose Date to base your strategy on a date field.</p> <p><b>Length:</b> Choose Length to base your strategy on the length of data in a field.</p> <p><b>Non_Blank:</b> Choose Non_Blank to base your strategy on the completeness of data in a field.</p> <p><b>Priority_Number:</b> Choose Priority_Number to base your strategy on a number.</p> <p><b>Priority_String:</b> Choose Priority_String to base your strategy on string data in a field.</p>
<b>Strategy priority</b>	<p>These are the choices for priorities for each of the best record strategies, other than Non_Blank and Custom.</p> <p>Date</p> <ul style="list-style-type: none"> <li>• <b>Newest:</b> The newest date in the field will cause an action to take place.</li> <li>• <b>Oldest:</b> The oldest date in a field will cause an action to take place.</li> </ul> <p>Length</p> <ul style="list-style-type: none"> <li>• <b>Longest:</b> The longest string in a field will cause an action to take place.</li> <li>• <b>Shortest:</b> The shortest string in a field will cause an action to take place.</li> </ul> <p>Priority_Number</p> <ul style="list-style-type: none"> <li>• <b>Highest:</b> The highest number in a field will cause an action to take place.</li> <li>• <b>Lowest:</b> The lowest number in a field will cause an action to take place.</li> </ul> <p>Priority_String</p> <ul style="list-style-type: none"> <li>• <b>Ascending:</b> The string with the most ascending string order will cause an action to take place.</li> <li>• <b>Descending:</b> The string with the most descending string order will cause an action to take place.</li> </ul>
<b>Strategy field</b>	Choose a field that contains data that you need to execute your strategy.

Option	Description
<b>Posting destination</b>	<p>Specifies the destination record.</p> <p><b>Master:</b> Post only to a master record.</p> <p><b>Subs:</b> Post only to subordinate records.</p> <p><b>Master to Subs:</b> Push information from the master record and post it to each subordinate record.</p> <p><b>All:</b> Post to both the master and subordinate records.</p>
<b>Post only once per destination</b>	<p><b>Yes:</b> Post only once per destination record. After data is posted to the destination record, the operation stops.</p> <p><b>No:</b> 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.</p> <p>Set this option to Yes when you are using the NON_BLANK strategy.</p> <p>Set this option to No when you are using the Longest, Shortest, Newest, Oldest, Ascending, or Descending priorities.</p> <p><b>Note:</b> This option is ignored when using the <b>Master to Subs</b> posting destination. With this posting destination, information can be posted only once.</p>
<b>View/Edit Python</b>	<p>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.</p>

### 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
<b>Source field</b>	Specifies the name of the source field in the input record.
<b>Destination field</b>	<p>Specifies the name of the destination field in the output record, or the destination of this best record action.</p> <p>You can have the action post to the same input field, or you can post to a different field.</p>

Option/Option group	Description
<b>Custom</b>	<p><b>Yes:</b> Specifies that you want to create custom Python code to perform an action on the destination field.</p> <p><b>No:</b> Specifies that you want to use the same source and destination fields.</p> <p>When this option is set to No, the contents of the source field are copied to the destination field.</p>
<b>Editor</b>	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 Topics

- [Designer Guide: Match, Best record](#)

### *Best record options: Destination Protection tab*

Protect data from being changed by enabling and defining destination protection.

Option	Description
<b>Best record name</b>	Enter a name for your Best Record operation. Make sure that this name is unique within this Match transform.
<b>Enable destination protection</b>	Select to protect records from best record operations that may modify the contents.
<b>Default destination protection</b>	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.
<b>Specify destination protection by field</b>	Select to enable destination protection through a value in a field.
<b>Destination protection field</b>	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.
<b>Specify destination protection by source</b>	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
<b>Source Name</b>	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.
<b>Destination protected</b>	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.

### *Group statistics options*

The Group Statistics option group includes the following options:

Option	Description
<b>Group statistics name</b>	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.
<b>Generate only basic statistics</b>	Select if you want to generate match group statistics. These will not include any statistics about input sources.
<b>Generate source statistics from input sources</b>	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.
<b>Generate source statistics from source values</b>	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.
<b>Logical source field</b>	Specifies the field that holds the value for your logical sources.
<b>Default logical source value</b>	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.
<b>Count all sources</b>	Select to count all sources, no matter what the value in the Logical source field is.
<b>Choose sources to count</b>	Select to specify particular sources to count, based on values in the Logical source field.

Option	Description
<b>Default count flag value</b>	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. <b>Yes:</b> Counts all of the records in the source. <b>No:</b> Does not count any of the records in the source.
<b>Auto generate sources</b>	Select to generate sources based on the value in a field.
<b>Predefined count flag field</b>	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
<b>Source value</b>	Specifies the value in the field to find. This value is case sensitive.
<b>Count</b>	Specifies whether you want to use the value you entered in the Logical source value option in the count. <b>Yes:</b> Includes the logical source value in the count. <b>No:</b> Does not include the value in the logical source value option in the count.

### 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
<b>Prioritization name</b>	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
<b>Input field</b>	Choose a field to sort your records on.
<b>Field order</b>	Specifies in which order records should be sorted.

### Group prioritization options: Record Completeness tab

Option	Description
<b>Prioritization name</b>	Specifies the name for this Group prioritization operation. If you have multiple operations in this Match transform, be sure to make this name unique.
<b>Order records based on completeness of data</b>	Select this option to apply priority and blank penalty points to records to help control the order of your records.
<b>Define only field penalties</b>	Select this option when penalties need to be assessed based on blank fields.
<b>Define priority and penalty fields</b>	Select this option when you have specific fields that contain the actual integer values for priority and blank penalty.
<b>Record priority field</b>	Choose the field that contains priority values. This field must contain an integer.
<b>Apply blank penalty field</b>	Choose the field that contains the indicator (Y or N) for applying blank penalty points to a record.
<b>Define priority and penalty based on input source</b>	Select to have your record priority and blank penalty indicator (Y or N) determined by membership in a given source.
<b>Source Name</b>	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.
<b>Priority</b>	Type a priority value (an integer) in the Priority column. Remember that the lower the priority score, the higher the priority.
<b>Apply Blank Penalty</b>	Choose Yes or No to determine whether a blank penalty is applied to a record based on membership to this source.

Option	Description
<b>Default record priority</b>	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.
<b>Default apply blank penalty</b>	<p>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 if a record does not belong to any of the sources specified.</p> <p><b>Yes:</b> 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.</p> <p><b>No:</b> No penalty is applied when the fields are blank.</p>
<b>Input field</b>	Displays the input fields available to assign a blank penalty score to.
<b>Blank penalty</b>	Assign a penalty value (an integer) to apply when the specified field is blank in a record.

### *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
<b>Unique ID name</b>	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.

Option	Description
<b>Processing operation</b>	<p>Specifies the type of processing operation you want the application to perform. Valid values include:</p> <p><b>Assign:</b> 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).</p> <p><b>AssignCombine:</b> 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).</p> <p><b>Combine:</b> 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).</p> <p><b>Delete:</b> Removes unique IDs from records that have one, unless they are protected.</p> <p><b>Split:</b> 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).</p>
<b>Recycle unique IDs</b>	<p>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:</p> <p><b>Yes:</b> Recycle freed-up unique IDs.</p> <p><b>No:</b> Do not recycle freed-up unique IDs.</p>
<b>ID field</b>	<p>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.</p>
<b>Field</b>	<p>The starting unique ID is obtained from an input field.</p> <p>Be sure to map in a field from an upstream transform before you add this option.</p>
<b>Starting unique ID field</b>	<p>Choose the field that passes in the starting unique ID. If no Unique ID is received, the starting number will default to 1.</p>
<b>Constant value</b>	<p>The starting ID is specified as a positive whole number in the Starting value option.</p>

Option	Description
<b>Starting value</b>	Indicates the starting unique ID value. Valid values range from 1 to UINT_MAX (unsigned integer max). The default value is 1.
<b>Value from file</b>	The starting Unique ID is read from the file specified in the File option.
<b>File</b>	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.
<b>GUID</b>	<p>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: <code>TimeLow-TimeMid-TimeHighAndVersion-ClockSeqAndReservedClockSeqLow-Node</code>.</p> <p>For more information about UUID, see the RFC document in the Related Topics section.</p>
<b>Save ending ID to file and reclaim recycled IDs</b>	<p>Specifies whether to save the last ID that was assigned to a file.</p> <p>Additionally, specifies whether to reclaim recycled IDs.</p>
<b>File</b>	Specifies the file to write the last assigned ID to.
<b>Allow multiple Match transforms to access unique ID file</b>	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 dataflow to share a single unique ID file.
<b>Number of IDs to get when accessing file</b>	<p>Specifies the number of IDs to retrieve from the unique ID file during each access.</p> <p>For example, with a setting of 100, the first process will access the file and retrieve IDs numbered 1-100. The next process will 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.</p> <p><b>Note:</b></p> <p>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.</p>

Option	Description
<b>Group number field</b>	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 Topics

- [Designer Guide: Match, Assign unique IDs using a file](#)
- [Designer Guide: Match, Unique ID](#)
- [UUID RFC: http://www.ietf.org/rfc/rfc4122.txt](http://www.ietf.org/rfc/rfc4122.txt)

### 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
<b>Unique ID name</b>	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.
<b>Enable destination protection</b>	Select if you want to protect a destination source from having its unique IDs overwritten with the IDs from matching records.
<b>Default destination protection</b>	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.
<b>Specify destination protection by field</b>	Select to enable destination protection through a value in a field.
<b>Unique ID protected field</b>	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.
<b>Specify destination protection by source</b>	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
<b>Source name</b>	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.
<b>Unique ID protected</b>	<b>Yes:</b> This source is protected. <b>No:</b> This source is not protected.

### *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
<b>Output flag selection name</b>	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: <ul style="list-style-type: none"> <li>• &lt;match level name&gt;_DMA_Matches_Select_Record</li> <li>• &lt;match level name&gt;_Mail_List_Select_Record</li> </ul>
<b>Unique</b>	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.
<b>Single source masters</b>	Highest ranking member of a match group whose members all came from the same source. Can be from normal- or special-type sources.
<b>Single source subordinates</b>	A record that came from a normal- or suppress-type source and is a subordinate member of a match group.
<b>Multiple source masters</b>	Highest ranking member of a match group whose members came from two or more sources. Can be from normal- or special-type sources.
<b>Multiple source subordinates</b>	A subordinate record of a match group that came from a normal- or suppress-type source whose members came from two or more sources.
<b>Suppression matches</b>	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.
<b>Suppression uniques</b>	Records that came from a suppress-type source, and for which no matching records were found.



Record type	Description
<b>Suppression masters</b>	A record that came from a suppress-type source and is the highest ranking member of a match group.
<b>Suppression subordinates</b>	A record that came from a suppress-type source and is a subordinate member of a match group.

### 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	<p>The distance used in proximity matching.</p> <p>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.</p> <p>&lt;logical-name&gt; 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.</p> <p>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.</p>

#### Related Topics

- [Designer Guide: Set up Geographic Proximity matching - Criteria options](#)

### 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
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 that made the decision ( if the Match_Type is <b>R</b> ). Otherwise, the field is blank.
Match_Level	None	Specifies the name of the match level used.
Match_Score	None	The Match_Score field outputs the following values: <ul style="list-style-type: none"> <li>• The criteria similarity score when the Match_Type is <b>R</b>.</li> <li>• The total weighted score when the Match_Type is <b>W</b>.</li> <li>• Blank if the record is a driver record (Match_Type of <b>D</b>) or if the records are unique.</li> </ul>
Match_Status	None	The values for the Match_Status field that appear in your output are: <p><b>D</b>: This record is a driver in a match group.</p> <p><b>P</b>: This record is a passenger in a match group.</p> <p><b>U</b>: This is a unique record.</p>
Match_Type	None	Describes how each record is identified as a match. Possible values are: <p><b>&lt;blank&gt;</b>: The record did not match any other record. It is a unique record.</p> <p><b>D</b>: The record was the driver record in the comparison process.</p> <p><b>R</b>: The record was identified as matching the driver record because one of the criteria met the Match_Score.</p> <p><b>W</b>: The record was identified as matching the driver because the total weighted score met the Weighted match score. See Matching methods in the <i>Data Services Designer Guide</i> for more information on weighted scores.</p>

### 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. <b>N:</b> The record comes from a normal source. <b>P:</b> The record comes from a suppress source. <b>S:</b> 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 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.

Field name	Description
Group_Source_Type	<p>Specifies the type of source in the match group. This field will contain one of the following values:</p> <p><b>M:</b> The records come from more than one input source (excluding records from Special sources).</p> <p><b>P:</b> 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.)</p> <p><b>S:</b> The records come from a single input source .</p> <p><b>&lt;empty&gt;:</b> The record is unique.</p>
Multi_Source_Count	<p>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.</p>
Source_Count	<p>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.</p>
Source_Group_Count	<p>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.</p>

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	<p>Provides the total number of records in the match group.</p> <p>Unique records have a value of 1.</p>
Group_Order	<p>The master record receives a value of 1. Subordinate records receive a value of 2 through the number of records in the match group.</p> <p>You may control the order by including a Group Prioritization in the Post Match Operations. Unique records have a value of 0.</p>

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	<p>Specifies whether a record contributed to the source count, and if so, whether there were other sources represented in the match group.</p> <p><b>M:</b> 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).</p> <p><b>S:</b> 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> <p><b>P:</b> 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.)</p>
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	<p>Specifies the type of logical source.</p> <p><b>N:</b> Normal source</p> <p><b>P:</b> Suppress source</p> <p><b>S:</b> Special source</p>

### 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 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
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.
Group_Rank	Specifies whether the record is a master (M) or a subordinate (S). Unique records have an empty value.

Field name	Description
Group_Type	<p>Specifies whether a record contributed to the source count, and if so, whether there were other sources represented in the match group.</p> <p><b>M:</b> 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).</p> <p><b>S:</b> 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> <p><b>P:</b> 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.)</p>
Source_Count	<p>Shows the number of logical sources in this match group.</p> <p>Unique records have a blank value</p>
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	<p>Specifies the type of logical source.</p> <p><b>N:</b> Normal source</p> <p><b>P:</b> Suppress source</p> <p><b>S:</b> Special source</p>

### 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	<p>Specifies the status of the Unique_ID output field. This field generates the following values:</p> <p><b>&lt;blank&gt;:</b> No change. The Unique_ID output field is the same as the Unique_ID input field.</p> <p><b>D:</b> Indicates that the Unique_ID output field has a blank unique ID and that the old unique ID was deleted.</p> <p><b>N:</b> Indicates that the Unique_ID output field has a new unique ID.</p> <p><b>O:</b> Indicates that the Unique_ID output field is assigned an old (existing) ID. This happens when a record is combined.</p>

Field name	Description
Record_Operation	<p>Specifies the operation that should be performed on the record, based on the input fields (except protected fields). This field generates the following values:</p> <p><b>&lt;blank&gt;</b>: Does not call an operation.</p> <p><b>A</b>: Assigns a unique ID to the record.</p> <p><b>C</b>: Combines the record's unique ID.</p> <p><b>D</b>: Deletes the record's unique ID.</p> <p><b>S</b>: Splits the record's unique ID.</p>
Unique_ID	<p>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.</p>

### 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	<p>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.</p>

### 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	<p>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.</p>



### 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 BusinessObjects 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 Topics**

- [Transform configurations](#)
- [Address Cleanse reference](#)

#### 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 Topics**

- [Transform configurations](#)
- [Downloading blueprints and other content objects](#)

## **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 Topics**

- [Report and analysis](#)
- [Reference files](#)
- [Transform performance](#)
- [USPS license information options](#)
- [NCOALink options](#)
- [Assignment options](#)
- [Standardization options](#)
- [Non Certified options](#)
- [CASS Report options](#)
- [Suggestion List group](#)
- [Z4 Change options](#)

## **5.4.13.3 Report and analysis**

Use these options to generate USA Regulatory Address Cleanse report data.

Option	Description
<b>Gather Statistics Per Data Source</b>	<p>Specifies whether to generate report data per Data_Source_ID field value.</p> <p><b>Yes:</b> Generates report statistics (if the <b>Generate Report Data</b> option is set to Yes) per Data_Source_ID field value.</p> <p><b>No:</b> Generates reports (if the <b>Generate Report Data</b> option is set to Yes) for the input database without generating statistics based on the Data_Source_ID field value.</p> <p><b>Note:</b> If you select Yes, other setup requirements apply. Read about statistics based on logical groups in the <i>Designer Guide</i>.</p>
<b>Generate Report Data</b>	<p>Specifies whether to generate report data for this transform.</p> <p><b>Yes:</b> Generates report data for this transform.</p> <p><b>No:</b> Turns off report data generation. If you do not need to generate reports (during testing, for example), you should set this option to <b>No</b> to improve performance.</p>

#### Related Topics

- [Designer Guide: Data Quality, Multiple data source statistics reporting](#)

### 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
<b>Address Directory 1</b>	<p>zip4us.dir</p> <p>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.</p> <p>Use the substitution variable \$\$RefFilesAddressCleanse.</p>

Option	Description
<b>Address Directory 2</b>	<p><code>*.dir</code></p> <p>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.</p>
<b>Address Geo 1 Directory</b> <b>Address Geo 2 Directory</b> <b>Address Geo 3 Directory</b> <b>Address Geo 4 Directory</b> <b>Address Geo 5 Directory</b> <b>Address Geo 6 Directory</b> <b>Address Geo 7 Directory</b> <b>Address Geo 8 Directory</b> <b>Address Geo 9 Directory</b> <b>Address Geo 10 Directory</b>	<p><code>ageo1.dir</code></p> <p>The Address-level GeoCensus files, <code>ageo1.dir</code> through <code>ageo10.dir</code>, are required only if you use the Address-level GeoCensus option or if you use the Geocoder transform.</p> <p>Use the substitution variable <code>\$\$RefFilesAddressCleanse</code>.</p>
<b>Address SHS Directory</b>	<p><code>zip4us.shs</code></p> <p>This directory enhances normal primary name lookups and is required for processing. Use the substitution variable <code>\$\$RefFilesAddressCleanse</code>.</p>
<b>Centroid Geo Directory</b>	<p><code>cgeo2.dir</code></p> <p>This directory is required only if you use the centroid-level GeoCensus option or if you use the Geocoder transform.</p> <p>Use the substitution variable <code>\$\$RefFilesAddressCleanse</code>.</p>
<b>City Directory</b>	<p><code>city##.dir</code></p> <p>The City directory is a table of city names, states, and ZIP Codes. It is organized by state and city. Use the substitution variable <code>\$\$RefFilesAddressCleanse</code>.</p>
<b>DPV Path</b>	<p>Specify the path to the DPV (Delivery Point Validation) directory files. These directory files are required for CASS certification. Use the substitution variable <code>\$\$RefFilesAddressCleanse</code>.</p>

Option	Description
<b>DSF2 Augment Path</b>	Specify the path to the directory that contains the DSF2 (Second Generation Delivery Sequence) files you received from the USPS. Use the substitution variable <code>\$\$RefFilesAddressCleanse</code> .
<b>eLOT Directory</b>	<code>elot.dir</code> 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 <b>Enable eLot</b> option in the Assignment Options group is set to <b>Yes</b> . Use the substitution variable <code>\$\$RefFilesAddressCleanse</code> .
<b>EWS Directory</b>	<code>ewsyymmdd.dir</code> The software lists <b>ew*.dir</b> in the Option Value column, so that the transform finds the most current directory. Use the substitution variable <code>\$\$RefFilesAddressCleanse</code> .
<b>LACSLink Path</b>	Specify the path to the LACSLink directory files. These directory files are required for CASS certification. Use the substitution variable <code>\$\$RefFilesAddressCleanse</code> .
<b>NCOALink Path</b>	The location for the NCOALink directory files. These directory files are required for NCOALink processing and certification. Use the substitution variable <code>\$\$RefFilesAddressCleanse</code> .
<b>Postcode Directory</b>	<code>zcf10.dir</code> This directory contains the same data as the City directory, but is organized by ZIP Code. Use the substitution variable <code>\$\$RefFilesAddressCleanse</code> .
<b>Postcode Reverse Directory</b>	<code>revzip4.dir</code> 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 <code>\$\$RefFilesAddressCleanse</code> .
<b>RDI Path</b>	The RDI directory indicates whether an address is residential. Use the substitution variable <code>\$\$RefFilesAddressCleanse</code> .

Option	Description
<b>Reverse Soundex Address Directory</b>	<p>zip4us.rev</p> <p>This directory enhances primary name lookups.</p> <p>Use the substitution variable <code>\$\$RefFilesAddressCleanse</code>.</p>
<b>SuiteLink Path</b>	<p>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.</p> <p>Specify the path to the SuiteLink directory files.</p> <p>Use the substitution variable <code>\$\$RefFilesAddressCleanse</code>.</p>
<b>USPS Log Path</b>	<p>Specify the path to the directory for NCOALink, DPV, and LACSLink log files. The provided substitution parameter is <code>\$\$CertificationLogPath</code>. The software determines the file names during processing as the USPS requires. This directory must already exist and be writable.</p> <p>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.</p>
<b>Z4 Change Directory</b>	<p>z4change.dir</p> <p>The Z4Change directory lists all the ZIP Codes and ZIP+4 Codes in the country.</p> <p>Use the substitution variable <code>\$\$RefFilesAddressCleanse</code>.</p>

For information about downloading directories, see the latest directories update.

#### Related Topics

- [Installation Guide: Additional Information, Directory data](#)

### 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, RDI, LACSLink, NCOALink, and SuiteLink processing.

Option	Description
<b>Cache DPV Directories</b>	<p>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.</p> <p><b>Yes:</b> Enables caching.</p> <p><b>No:</b> Disables caching.</p>
<b>Cache DSF2 Augment Directories</b>	<p>Specifies whether the DSF2 Augment directories are cached into memory.</p> <p><b>Yes:</b> Enables caching.</p> <p><b>No:</b> Disables caching.</p>
<b>Cache LACSLink Directories</b>	<p>Specifies whether the LACSLink directories are cached into memory.</p> <p><b>Yes:</b> Enables caching.</p> <p><b>No:</b> Disables caching.</p>
<b>Cache RDI Directories</b>	<p>Specifies whether the RDI directories are cached into memory.</p> <p><b>Yes:</b> Enables caching.</p> <p><b>No:</b> Disables caching.</p>
<b>Cache SuiteLink Directories</b>	<p>Specifies whether the SuiteLink directories are cached into memory.</p> <p><b>Yes:</b> Enables caching.</p> <p><b>No:</b> Disables caching.</p>
<b>Insufficient Cache Memory Action</b>	<p>Specifies the action to take if there is insufficient memory to cache the directories that you have set up for caching.</p> <p><b>Continue:</b> Attempts to continue initialization without caching.</p> <p><b>Error:</b> Issues an error and terminates the transform.</p>

Option	Description
<b>NCOALink Caching Mode</b>	<p>Specifies the method for caching NCOALink directories.</p> <p><b>Auto:</b> Select to have Data Services use available memory for caching.</p> <p><b>Manual:</b> 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 <b>NCOALink Memory Usage</b> option.</p> <p><b>None:</b> 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.</p>
<b>NCOALink Memory Usage</b>	<p>If the <b>NCOALink Caching Mode</b> 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.</p> <p><b>Note:</b> If the <b>Degree of Parallelism</b> value is greater than one, the <b>NCOALink Memory Usage</b> value is the total to be allocated for all threads. The value is not per thread.</p>

#### 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
<b>Customer Company Name</b> <b>Customer Company Address</b> <b>Customer Company Locality</b> <b>Customer Company Region</b> <b>Customer Company Postcode1</b> <b>Customer Company Postcode2</b>	<p>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.</p> <p>The customer information appears in the NCOALink Processing Summary report and log files.</p> <p>The provided substitution parameters for these fields are:</p> <p>\$\$CompanyName          \$\$CompanyAddress          \$\$CompanyLocality          \$\$CompanyRegion          \$\$CompanyPostcode1          \$\$CompanyPostcode2</p>
<b>Customer Company Phone</b>	<p>This is an optional field. The provided substitution parameter for this field is \$\$CompanyPhone.</p>
<b>DSF2 Licensee ID</b>	<p>Enter your DSF2 identification number as the USPS assigned it to you.</p> <p>You can use the substitution variable \$\$DSF2LicenseeID in this option.</p>
<b>IMB Mailer ID</b>	<p>This is an optional field.</p> <p>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.</p> <p>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).</p>
<b>Licensee Name</b>	<p>This field is required for NCOALink.</p> <p>The name of the company, up to 30 characters, as mentioned in the license agreement with the USPS. The licensee performs the NCOALink processing.</p> <p>This information appears in the PAF log and NCOALink Processing Summary report.</p>
<b>List Owner NAICS Code</b>	<p>This is a required field.</p> <p>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 <a href="http://www.census.gov/epcd/www/naics.html">http://www.census.gov/epcd/www/naics.html</a>.</p>

Option	Description
<b>List ID</b>	<p>The Customer or List ID is required for NCOALink limited and full service providers. End users may leave it blank.</p> <p>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:</p> <ul style="list-style-type: none"> <li>• First 3 digits: Customer name/identifier</li> <li>• Last 3 digits: List name identifier</li> </ul>
<b>List Processing Frequency</b>	<p>This 2-digit number (from 1 to 52) indicates how many times per year the list is processed with NCOALink.</p> <p>If the list owner has other lists processed by the NCOALink licensee at different frequencies, enter 99.</p>
<b>List Received Date</b>	Enter the date when the NCOALink licensee received the list. Use the <code>yyyy/mm/dd</code> format. If you are an end user, you may leave this blank.
<b>List Return Date</b>	Enter the date when the list will be returned to the customer. Use the <code>yyyy/mm/dd</code> format. If you are an NCOALink end user, you may leave this blank.
<b>Provider Level</b>	<p>This option lists the provider levels for which you have a registered license keycode. It defaults to the substitution parameter <code>\$\$USPSProviderLevel</code>.</p> <p>Only provider levels supported in your registered keycodes display in the option list.</p>

#### 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 <sup>1</sup>	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		

Option	NCOALink Full Service Provider*	NCOALink Limited Service Provider <sup>1</sup>	NCOALink End User*	DSF2	DPV	LACSLink
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 Post-code1	X	X	X	X	X	X
Customer Company Post-code2	X	X	X	X	X	X
Customer Company Phone						
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		

### 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
<b>Mailing List Name</b>	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.
<b>Platform ID</b>	The platform ID is the NCOALink licensee's identification number that is assigned by the USPS. It's exactly four characters long.

#### Related Topics

- [Processing options](#)
- [Report Options](#)
- [NCOALink Output Options](#)
- [Processing Acknowledgment Form \(PAF\) Details](#)
- [Service Provider Options](#)
- [Contact Detail List](#)

#### 5.4.13.7.1 Processing options

The following table describes the **NCOALink Processing Options**.

Option	Description
<b>Consider Moves Within Months</b>	<p>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.</p> <p>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.</p>
<b>External Processes Updating List</b>	Indicate whether the list undergoes additional processing before or after the USA Regulatory Address Cleanse transform.

Option	Description
<b>High Match Rate Expectancy</b>	<p>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 <b>None</b> or leave blank if you don't expect a high match rate. This option provides legitimate reasons for a high match rate.</p> <p><b>None:</b> Default.</p> <p><b>ANKLink Processed List:</b> 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.</p> <p><b>Stage File:</b> If you're performing Stage I or Stage II testing, ensure that the <b>List Processing Objective</b> is set to a Stage option.</p> <p><b>Return Mail List:</b> A returned mail list file contains records for mail that was returned to sender.</p>
<b>List Processing Mode</b>	<p><b>Change Of Address:</b> You're processing this job to update it with the latest address data. Default.</p> <p><b>Statistics Only:</b> 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.</p> <p><b>Return Codes Only:</b> 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.</p>

Option	Description
<b>List Processing Objective</b>	<p>Specify your reason for using NCOALink:</p> <p><b>Employee Training:</b> You're processing this file as part of employee training.</p> <p><b>Internal Database Testing:</b> You're testing with a licensee-owned database.</p> <p><b>Marketing:</b> You're testing with external customer lists.</p> <p><b>Normal:</b> You're processing the mailing list to update it before a mailing. Default.</p> <p><b>Stage I and Stage II:</b> 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.</p> <p><b>System Testing:</b> You're processing this file as part of system testing such as loading USPS file updates.</p> <p><b>Note:</b> When certifying for CASS and DSF2, you indicate the reason in the <b>Assignment Options &gt; USPS Certification Testing Mode</b> option.</p>
<b>Processing First Class Mail</b> <b>Processing Periodicals</b> <b>Processing Standard Mail</b> <b>Processing Package Service Mail</b>	<p>Select the types of mail to process by selecting <b>Yes</b> or <b>No</b> for each option.</p> <p><b>Processing First Class Mail</b> defaults to <b>Yes</b>; the others default to <b>No</b>.</p>
<b>Retrieve Move Types</b>	<p>Choose the types of moves to process:</p> <p><b>Business:</b> Business moves only.</p> <p><b>Individual:</b> Individual moves only.</p> <p><b>Individual and Business</b></p> <p><b>Individual and Family</b></p> <p><b>Individual and Family and Business:</b> Default.</p>

#### 5.4.13.7.2 Report Options

There is one option in the **NCOALink Report Options** group.

Option	Description
<b>Generate Return Code Descriptions</b>	<p>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.</p> <p><b>Yes:</b> Includes the report codes in the NCOALink Processing Summary report.</p> <p><b>No:</b> Default. Excludes report codes from the NCOALink Processing Summary report.</p>

#### 5.4.13.7.3 NCOALink Output Options

There is one option in the **NCOALink Output Options** group.

Option	Description
<b>Apply Move to Standardized Fields</b>	<p>Component output fields are not affected by this option.</p> <p><b>Yes:</b> Default. Data Services updates standardized fields to contain details about the address available through NCOALink.</p> <p><b>No:</b> Standardized output fields have the standardized version of input rather than the moved address.</p>

#### 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
<b>Company Website</b>	Enter the company website address for the person signing the PAF. You can leave this parameter blank.
<b>Customer Alternate Company Name</b>	If the list owner's company is also known by another name, enter that alternate name here.
<b>Customer Parent Company Name</b>	If the list owner's company is owned by another company, enter the parent company's name here.
<b>Date Signed By Licensee</b>	Enter the date that the licensee (the NCOALink service provider) signed the PAF in yyyy/mm/dd format.

Option	Description
<b>Date Signed By Customer</b>	Enter the date the customer signed the PAF in yyyy/mm/dd format.
<b>Email of Person Signing</b>	Enter the email address for the person who is signing the PAF. You can leave this parameter blank.
<b>Name Of Person Signing</b>	Enter the name of the person signing this PAF, up to 50 characters.
<b>Title Of Person Signing</b>	Enter the job title of the person signing this PAF, up to 50 characters.
<b>Type</b>	<p><b>Initial:</b> This is the first PAF you're completing to become authorized to process addresses for this particular customer.</p> <p><b>Modified:</b> You're completing a new PAF because some information on your old one changed.</p> <p><b>Renewal:</b> You're completing a new PAF because your old one is expiring.</p>
<b>Using Alternative PAF</b>	<p><b>Yes:</b> Select if you are using a PAF that is not the USPS form (you must have permission from the USPS).</p> <p><b>No:</b> The default setting for this field.</p> <p>This field requires either a <b>Yes</b> or <b>No</b>.</p>
<b>Using Cooperative Database</b>	<p>Indicates whether the list is from a cooperative database. Applicable for Full and Limited Service Providers only.</p> <p><b>Yes</b></p> <p><b>No:</b> (the default setting)</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>A PAF must be on file for each participant in the cooperative database.</li> <li>When set to Yes, a "C" is included in the PAF log to indicate that the list processed was a cooperative database.</li> </ul>



#### 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
<b>Additional Notes</b>	<p><b>None:</b> Default.</p> <p><b>Customer Requested Extension:</b> Select if the customer submitted a written request for an extension.</p>
<b>Buyer Company Name</b>	If the list was processed for rent, sale, or lease, enter the name of the company or individual who bought the list.
<b>Concurrent Processed Data Modified</b>	<p><b>No:</b> 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.</p> <p><b>From Postal Data:</b> If you are processing this data in some other way while performing NCOALink processing, indicates whether the processing includes changes with postal data.</p> <p><b>From Non Postal Data:</b> If you are processing this data in some other way while performing NCOALink processing, indicates whether the processing includes changes with non-postal data.</p> <p><b>From Both:</b> 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.</p>
<b>Concurrent Processes Performed</b>	<p>Indicates whether you processed or will process this data in some other way while performing NCOALink processing.</p> <p><b>Yes</b></p> <p><b>No</b></p>
<b>In House List Processing</b>	<p>Indicates whether the list is an in-house (internal) list. Applicable for Full Service Providers only.</p> <p><b>Yes</b></p> <p><b>No</b></p> <p><b>Note:</b> When set to Yes, an “I” is included in the Customer Service log to indicate that the list was an in-house list.</p>

Option	Description
<b>Output Returned</b>	<p>Identifies the type of output returned to the client.</p> <p><b>Standard:</b> All required NCOALink output was returned to the client. Default.</p> <p><b>Modified:</b> One or more post processes modified the return information (updates were applied to the list).</p> <p><b>Both:</b> 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.</p>
<b>Post Processed Data Modified</b>	<p><b>No:</b> If you are processing this data after performing NCOALink processing, indicates that the postprocessing does not include changes to the data. Default.</p> <p><b>From Postal Data:</b> If you are processing this data after performing NCOALink processing, indicates whether the postprocessing includes changes with postal data.</p> <p><b>From Non Postal Data:</b> If you are processing this data after performing NCOALink processing, indicates whether the postprocessing includes changes with non-postal data.</p> <p><b>From Both:</b> If you are processing this data after performing NCOALink processing, indicates that the postprocessing includes changes with both postal and non-postal data.</p>
<b>Post Processes Performed</b>	<p>Indicates whether you are processing this data after performing NCOALink processing.</p> <p><b>Yes</b></p> <p><b>No</b></p>
<b>Postcode For Mail Entry</b>	<p>Specifies the ZIP Code of the Business Mail Entry Unit (BMEU) or post office where the mail will be submitted for mailing.</p>

Option	Description
<b>Pre Processed Data Modified</b>	<p><b>No:</b> If you are processing this data before performing NCOALink processing, indicates that the preprocessing does not include changes to the data. Default.</p> <p><b>From Postal Data:</b> If you are processing this data before performing NCOALink processing, indicates whether the preprocessing includes changes with postal data.</p> <p><b>From Non Postal Data:</b> If you are processing this data before performing NCOALink processing, indicates whether the preprocessing includes changes with non-postal data.</p> <p><b>From Both:</b> If you are processing this data before performing NCOALink processing, indicates that the preprocessing includes changes with both postal and non-postal data.</p>
<b>Pre Processes Performed</b>	<p>Indicate whether you processed or will process this data before performing NCOALink processing.</p> <p><b>Yes</b></p> <p><b>No</b></p>

#### 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
<b>Address</b>	Enter the broker's or list administrator's address.
<b>Contact Level</b>	<p>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.</p> <p>Note that the transform doesn't use this value in any logs.</p>
<b>Contact Company Website</b>	Enter the website of the broker or list administrator. You can leave this parameter blank.
<b>Locality</b>	Enter the broker's or list administrator's locality (city).

Option	Description
<b>Type</b>	<p><b>Broker:</b> A broker directs business to the service provider.</p> <p><b>List Administrator:</b> A list administrator stores and maintains address lists.</p>
<b>License Assigned ID</b>	Enter a unique six-character ID number for the broker or list administrator. You assign the ID number.
<b>NAICS Code</b>	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 <a href="http://www.census.gov/epcd/www/naics.html">http://www.census.gov/epcd/www/naics.html</a> .
<b>Name</b>	Enter the broker's or list administrator's name.
<b>PAF Signing Date</b>	Enter the date when the contact signed the PAF in the format yyyy/mm/dd.
<b>Phone</b>	Enter the broker's or list administrator's phone number.
<b>Postcode1</b>	Enter the broker's or list administrator's Postcode1 (ZIP Code).
<b>Postcode2</b>	Enter the broker's or list administrator's Postcode2 (ZIP+4 Code).
<b>Region</b>	Enter the broker's or list administrator's region (state).

### 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
<b>Dual Address</b>	<p>Specifies the action to take when the transform encounters a dual address.</p> <p><b>Position:</b> 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.</p> <p><b>Postal:</b> The transform attempts to validate based on the postal address. If that fails, the transform attempts again based on the street address.</p> <p><b>Street:</b> The transform attempts to validate based on the street address. If that fails, the transform attempts again based on the postal address.</p>
<b>Enable DPV</b>	<p>Specifies whether to perform DPV (Delivery Point Validation) processing.</p> <p><b>Yes:</b> Enables DPV processing.</p> <p><b>No:</b> Disables DPV processing.</p>
<b>Enable DSF2 Augment</b>	<p>Specifies whether to perform DSF2 (Delivery Sequence File Second Generation) augment processing.</p> <p><b>Yes:</b> Enables DSF2 augment processing.</p> <p><b>No:</b> Disables DSF2 augment processing.</p>
<b>Enable eLOT</b>	<p>Specifies whether to perform eLOT (Enhanced Line of Travel) processing.</p> <p><b>Yes:</b> Enables eLOT processing.</p> <p><b>No:</b> Disables eLOT processing.</p>
<b>Enable EWS</b>	<p>Specifies whether to perform EWS (Early Warning System) processing.</p> <p>If this transform cannot make an exact match within the <code>zip4us.dir</code> (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.</p> <p><b>Yes:</b> Enables EWS processing.</p> <p><b>No:</b> Disables EWS processing.</p>

Option	Description
<b>Enable LACSLink</b>	<p>Specifies whether to perform LACSLink (Locatable Address Conversion System) processing.</p> <p><b>Yes:</b> Enables LACSLink processing.</p> <p><b>No:</b> Disables LACSLink processing.</p>
<b>Enable NCOALink</b>	<p>Specifies whether to perform NCOALink (National Change of Address) processing.</p> <p><b>Yes:</b> Enables NCOALink processing.</p> <p><b>No:</b> Disables NCOALink processing.</p>
<b>Enable RDI</b>	<p>Specifies whether to perform RDI (Residential Delivery Indicator) processing.</p> <p><b>Yes:</b> Enables RDI processing.</p> <p><b>No:</b> Disables RDI processing.</p>
<b>Enable Reverse Soundex Search</b>	<p>Specifies whether to use the <code>zip4us.rev</code> (Reverse Soundex) directory in an attempt to make address assignments.</p> <p><b>Yes:</b> Enables Reverse Soundex.</p> <p><b>No:</b> Disables Reverse Soundex.</p>
<b>Enable SuiteLink</b>	<p>Specifies whether to perform SuiteLink processing.</p> <p><b>Yes:</b> Enables SuiteLink processing.</p> <p><b>No:</b> Disables SuiteLink processing.</p>

Option	Description
<b>Geo Mode</b>	<p>Specifies which type of GeoCensus processing you want to perform.</p> <p><b>Address:</b> The transform processes address-level GeoCensus only.</p> <p><b>Both:</b> 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.</p> <p><b>Centroid:</b> The transform processes centroid-level GeoCensus only.</p> <p><b>None:</b> 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.</p> <p><b>Note:</b> 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 <i>Upgrade Guide</i>.</p>
<b>USPS Certification Testing Mode</b>	<p>Indicates the type of certification being processed so that the software checks for the appropriate settings and issues warnings and errors when applicable.</p> <p><b>None:</b> The transform processes the job normally, without any special settings for certification. This is the default setting.</p> <p><b>CASS:</b> The transform processes the job with the appropriate settings for CASS self-certification.</p> <p><b>DSF2 Augment:</b> The transform processes the job with appropriate settings for DSF2 augment certification.</p> <p><b>Note:</b> When certifying for NCOALink, you set the testing mode in the <b>NCOALink &gt; Processing Options &gt; List Processing Objective</b> option.</p>

### 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
<b>Add Firm Match Secondary</b>	<p><b>Yes:</b> Adds secondary address information obtained from SuiteLink directories to the address line.</p> <p><b>No:</b> 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.</p> <p>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.</p> <p>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.</p>
<b>Address Line Alias</b>	<p>Specifies how to standardize the address line if the input primary address is an alias.</p> <p><b>Convert:</b> Converts address lines to the preferred form found in the postal directory.</p> <p><b>Preserve:</b> Retains address lines as they were input.</p> <p><b>Note:</b> To be compliant with CASS, set up your jobs to return the USPS preferred address. When the <b>Address Line Alias</b> option is set to <b>Convert</b>, 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 (<b>Address Line Alias</b> set to <b>Preserve</b>), but the software does not produce a USPS 3553 form.</p>
<b>Append Private Mailbox</b>	<p>Private mailboxes (PMB) are like post office boxes, except that they are hosted by private companies.</p> <p><b>Yes:</b> Places the address and PMB in the same field.</p> <p><b>No:</b> 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.</p>



Option	Description
<b>Assign With Input Locality</b>	<p>Specifies whether to use the last-line index when assigning the locality (city) name.</p> <p><b>Yes:</b> 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.</p> <p><b>No:</b> 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 <b>No</b>, the value you choose for the <b>Preserve Place Names</b> option does not matter; place names are converted.</p> <p><b>Note:</b> When the <b>Use USPS Locality Abbreviation</b> option is set to <b>Yes</b>, <b>Preserve Place Names</b> is set to <b>Yes</b>, and <b>Assign with Input Locality</b> is set to <b>No</b>, the software may not preserve some place names over 13 characters and abbreviates them.</p>
<b>Capitalization</b>	<p>Specifies the casing of your address data.</p> <p><b>Lower:</b> Converts data to all lowercase letters. For example, "Main Street South" becomes "main street south."</p> <p><b>Mixed:</b> Converts data to initial capital letters. For example, "MAIN STREET SOUTH" becomes "Main Street South."</p> <p><b>Upper:</b> Converts data to all capital letters. For example, "Main Street South" becomes "MAIN STREET SOUTH."</p> <p><b>Note:</b> If you want consistent casing for your data, make sure that this option and the <b>Capitalization</b> setting in the Data Cleanse transform are the same.</p>
<b>Combine Multi-lines</b>	<p>Specifies what to do with related fields input on separate lines.</p> <p><b>Yes:</b> Looks for related fields that were input on separate lines, and tries to put them together on the same line.</p> <p><b>No:</b> Does not try to combine fields.</p>
<b>Directional Style</b>	<p>Specifies whether to abbreviate directional data.</p> <p><b>Long:</b> Uses fully-spelled directionals such as "North," "South," "East," and "West."</p> <p><b>Preserve:</b> Preserves the style used in the input record.</p> <p><b>Short:</b> Uses abbreviated directionals such as "N," "S," "E," and "W."</p> <p><b>Note:</b> When the <b>Use USPS Street Abbreviation</b> option is set to <b>Yes</b>, the software overrides a setting of <b>Long</b> and <b>Preserve</b> for the <b>Directional Style</b> option and outputs the short directional style.</p>

Option	Description									
Include Unused Address Line Data	Specifies whether to output the unused address line data (for discrete and multiline fields).  <b>Yes:</b> Includes the unused address line data, including invalid secondary information, in the address line and in the ADDRESS_LINE_REMAINDER1 field.  <b>No:</b> 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></table>	Input	Include Unused Address Line Data option			Yes	No	SAP 332 FRONT ST S FL 9 LA CROSSE WI 54601	SAP 332 FRONT ST S FL 3 FL 9 LA CROSSE WI 54601-4025	SAP 332 FRONT ST S FL 3 LA CROSSE WI 54601-4025
	Input	Include Unused Address Line Data option								
		Yes	No							
	SAP 332 FRONT ST S FL 9 LA CROSSE WI 54601	SAP 332 FRONT ST S FL 3 FL 9 LA CROSSE WI 54601-4025	SAP 332 FRONT ST S FL 3 LA CROSSE WI 54601-4025							
	Output fields									
	Primary_Secondary_Address	332 FRONT ST S FL 3	332 FRONT ST S FL 3							
	Full_Address	332 FRONT ST S FL 3 FL 9	332 FRONT ST S FL 3							
	Address_Line_Remainder1	FL 9	FL 9							
	Secondary_Address	FL 3	FL 3							
	Pre_SuiteLink_Unit_Description	FL	FL							
Pre_SuiteLink_Unit_Number	9	9								

Option	Description
<b>Move Multiline Data</b>	<p>If you turn on this feature, the <b>USA Regulatory Address Cleanse</b> transform rearranges your multiline data to conform to USPS guidelines.</p> <p>The transform moves the primary address into position above the locality-region-postal code line (or lastline).</p> <p><b>Bottom:</b> 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.</p> <p><b>No:</b> Does not rearrange any lines, blank or otherwise.</p> <p><b>Top:</b> 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.</p> <p>Example:</p> <div data-bbox="376 751 938 978" data-label="Diagram"> <pre> graph LR     subgraph Input_data [Input data:]         L1[Line1: 100 Market Street]         L2[Line2: Suite 202]         L3[Line3: Sycamore Building]         L4[Line4: ]         L5[Line5: Boston MA 02109]     end     subgraph Result [Result of moving:]         R1[Sycamore Building]         R2[Suite 202]         R3[100 Market St]         R4[Boston MA 02109]     end     L3 --&gt; R1     L2 --&gt; R2     L1 --&gt; R3     L5 --&gt; R4 </pre> </div> <p>This feature does not require that you standardize your data.</p> <p><b>Note:</b> If you choose <b>Top</b> or <b>Bottom</b>, the input field lengths of all fields mapped to Multiline<sub>x</sub> must be the same. For example, if Multiline<sub>1</sub> is set to a length of 60, all Multiline fields that you use must be set to 60.</p>
<b>Multiline Update Postcode 1</b>	<p>This option affects multiline data only (data passed in and retrieved through multiline fields).</p> <p><b>Dont_Update:</b> 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.</p> <p><b>Erase_Then_Update:</b> 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.</p> <p><b>Update:</b> Replaces the input Postcode1 with the assigned Postcode1 in the Multiline output fields. If no Postcode1 is assigned, the original is retained.</p>

Option	Description
<b>Multiline Update Postcode 2</b>	<p>This option affects multiline data only (data passed in and retrieved through multiline fields).</p> <p><b>Dont_Update:</b> 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.</p> <p><b>Erase_Then_Update:</b> 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.</p> <p><b>Update:</b> In the Multiline output fields, replaces the input Postcode2 with the assigned Postcode2. If no Postcode2 is assigned, it retains the original.</p>
<b>Preserve Dual Address Order</b>	<p>Specifies whether to preserve or change the dual address order.</p> <p><b>Yes:</b> When an address contains both a street and mailing address, keeps the address order as it was input.</p> <p><b>No:</b> When the input address contains both a locality and mailing address, moves the assigned address immediately above the locality and region.</p>
<b>Preserve Place Names</b>	<p>Specifies whether to preserve or change non-mailing city names (place names).</p> <p><b>Yes:</b> Preserves the non-mailing city name. Given Hollywood as input, the transform outputs Hollywood.</p> <p><b>No:</b> Changes non-mailing city names to city names preferred by the USPS. Given Hollywood as input, the transform outputs Los Angeles.</p> <p><b>Note:</b> When the <b>Use USPS Locality Abbreviation</b> option is set to <b>Yes</b>, <b>Preserve Place Names</b> is set to <b>Yes</b>, and <b>Assign with Input Locality</b> is set to <b>No</b>, the software may not preserve some place names over 13 characters and abbreviates them.</p>
<b>Primary Type Style</b>	<p>Specifies whether to abbreviate the street (primary) type.</p> <p><b>Long::</b> Uses fully-spelled primary types such as Street, Avenue, and Road.</p> <p><b>Preserve:</b> Preserves the style used in the input record.</p> <p><b>Short:</b> Uses abbreviated primary types such as St, Ave, and Rd.</p> <p><b>Note:</b> When <b>Use USPS Street Abbreviation</b> is set to <b>Yes</b>, the software overrides a setting of <b>Long</b> and <b>Preserve</b> for the <b>Primary Type Style</b> option and outputs the short primary type style.</p>

Option	Description
<b>Retain Pound Sign in Unit Description</b>	<p>Outputs “#” into either extraneous fields or to the UNIT_DESCRIPTION output field.</p> <p><b>Yes:</b> Outputs the # unit designator to the UNIT_DESCRIPTION output field.</p> <p><b>No:</b> Outputs the # unit designator to the EXTRANEIOUS_SECONDARY_UNIT_NUMBER and/or the EXTRANEIOUS_SECONDARY_ADDRESS_DATA output fields.</p> <p>The option does not affect the following address situations:</p> <ul style="list-style-type: none"> <li>• Puerto Rican addresses</li> <li>• Military addresses</li> <li>• Rural Route addresses</li> <li>• Addresses without “#” in the address line</li> <li>• Addresses with remainder words</li> </ul>
<b>Standardize Assigned Address</b>	<p>Specifies whether to correct and standardize the assigned address line and lastline data.</p> <p><b>Yes:</b> Corrects and standardizes your address line and lastline data. Use this value for CASS certification.</p> <p><b>No:</b> Does not standardize your address line or lastline data.</p>
<b>Standardize Unassigned Address</b>	<p>Specifies whether to standardize unassigned data.</p> <p><b>Yes:</b> Attempts to parse and standardize any unassigned addresses.</p> <p><b>No:</b> Leaves unassigned addresses as entered on input.</p>
<b>Unit Description</b>	<p>Specifies how to standardize the unit description.</p> <p><b>Convert:</b> Uses the unit description found in the postal directory (such as an apartment, suite, room, or floor).</p> <p><b>Preserve:</b> Preserves the unit description from the input record and corrects any spelling errors.</p>

Option	Description
<b>Use USPS Locality Abbreviation</b>	<p><b>Yes:</b> Enables this option (and affects only the multiline and standardized last line fields).</p> <p><b>No:</b> Disables this option.</p> <p>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.</p> <p>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.</p> <p><b>Note:</b></p> <p>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.</p> <p>Intelligent truncation abbreviates the output data first, and if it still doesn't fit the output buffer, it truncates the data.</p> <p>There are no options to set this up in the USA Regulatory Address Cleanse transform. The transform does this automatically.</p> <p>If the <b>Use USPS Primary Name Abbreviation</b> and/or the <b>Use USPS Locality Abbreviation</b> 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.</p>

Option	Description
<b>Use USPS Primary Name Abbreviation</b>	<p><b>Yes:</b> Enables this option.</p> <p><b>No:</b> Disables this option.</p> <p>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.</p> <p><b>Note:</b> 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.</p> <p>This option affects multiline and standardized last line fields when set to <b>Yes</b>.</p> <p>If the <b>Use USPS Street Abbreviation</b> option is set to <b>Yes</b>, it affects the following address components on output:</p> <ul style="list-style-type: none"> <li>• <b>Suffix Style:</b> The style will be short.</li> <li>• <b>Directional Style:</b> The style will be short.</li> <li>• <b>Address Line Alias:</b> The setting of <b>Preserve</b> may be overridden.</li> </ul> <p><b>Note:</b> 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.</p> <p>Intelligent truncation abbreviates the output data first, and if it still doesn't fit the output buffer, it truncates the data.</p> <p>There are no options to set this up in the USA Regulatory Address Cleanse transform. The transform does this automatically.</p> <p>If the <b>Use USPS Primary Name Abbreviation</b> and/or the <b>Use USPS Locality Abbreviation</b> 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.</p>

### 5.4.13.10 Non Certified options

This option group includes options to process your data without following CASS certification rules.

Option	Description
<b>Accept Inexact Postcode Move</b>	<p>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.</p> <p><b>Yes:</b> Ignores the non-matching elements and uses built-in matching thresholds.</p> <p><b>No:</b> Disables this option (and does not ignore non-matching elements).</p>
<b>Assign With Input Postcode</b>	<p>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.</p> <p><b>Yes:</b> 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.</p> <p><b>No:</b> Disables this option.</p>
<b>Assign Postcode2 Not DPV Validated</b>	<p>Specifies whether to output the Postcode2 when an assignment is made even when <b>Enable DPV</b> is set to <b>No</b> and <b>Disable Certification</b> is set to <b>Yes</b>. The output address is not validated by DPV.</p> <p><b>Yes:</b> Assigns the Postcode2 when <b>Enable DPV</b> is set to <b>No</b>.</p> <p><b>No:</b> Leaves the Postcode2 blank when an assignment is made and <b>Enable DPV</b> is set to <b>No</b>.</p>
<b>Disable Certification</b>	<p>Specifies whether to run address cleansing for CASS certification.</p> <p><b>Yes:</b> 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.</p> <p><b>No:</b> Runs address cleansing under CASS certification rules. The other options in this option group are ignored.</p>



Option	Description
<b>Enable Geo Only</b>	<p><b>Note:</b> 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 <i>Upgrade Guide</i>.</p> <p>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.</p> <p><b>Yes:</b> 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.</p> <p><b>No:</b> Enables you to run address cleansing with the GeoCensus directories, as well as the postal directories.</p>
<b>Enable Parse Only</b>	<p>Specifies whether the transform should parse and validate your data or parse only.</p> <p><b>Yes:</b> 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.</p> <p><b>No:</b> Parses records into their discrete components and performs a lookup in the postal directories. This mode may be slower, but parsing results are verified.</p>
<b>Enable Suggestion Lists</b>	<p>Specifies whether suggestion lists are generated for records that cannot be assigned. This option is for transactional projects.</p> <p><b>Yes:</b> Generates suggestion lists.</p> <p><b>No:</b> Does not generate suggestion lists.</p>

### 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
<b>Company Name Certified</b>	If you rely on SAP BusinessObjects for vendor CASS certification, leave this parameter blank; the transform inserts "SAP BusinessObjects" as the default value. If you have your own end-user CASS certification from the USPS, type your company name (up to 40 characters).
<b>List Name</b>	Specifies the name of the mailing list (up to 20 characters).
<b>List Owner</b>	Specifies the name of your company (up to 19 characters).
<b>LOT Certification</b>	<p>Specifies whether you have LOT certification.</p> <p><b>Yes:</b> You have LOT certification but you do not have CASS certification in your own name.</p> <p><b>No:</b> You have CASS certification in your own name but you did not seek or obtain LOT certification.</p> <p>In this case, setting LOT Certified to No ensures that the LOT Certification lines on your USPS 3553 forms are blank, which is appropriate.</p>
<b>Mailer Address 1</b>	Specifies the name and address of the person or organization for whom you are preparing the mailing (up to 29 characters per line).
<b>Mailer Address 2</b>	
<b>Mailer Address 3</b>	
<b>Mailer Address 4</b>	
<b>Software Version</b>	<p>If you rely on SAP BusinessObjects for vendor CASS certification, you may leave this parameter blank. The transform inserts the appropriate software name and version as the default value.</p> <p>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.</p>

### 5.4.13.12 Suggestion List group

Set the options in this group to configure how suggestion lists are output.

Option/Option group	Description
<b>Address Lines Match Minimum</b>	<p>Specifies the similarity score required for address-line suggestions. Valid values are 0 to 80.</p> <p>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.</p>
<b>Address Range Window</b>	<p>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.</p> <p>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:</p> <p>1000 Pine St.</p> <p>Suggestions would only be returned in a range from 750 to 1250 Pine Street.</p> <p>Type "0" if you don't want to limit the ranges returned in suggestions.</p>
<b>Combine Overlapping Ranges</b>	<p>Specifies how individual suggestions with overlapping ranges are consolidated.</p> <p><b>Combine_Ignoring_Gaps:</b> Ignores gaps and overlaps in primary ranges, so consolidation is more aggressive.</p> <p><b>Combine_Preserving_Gaps:</b> Preserves gaps in primary ranges, but overlapping ranges are consolidated.</p> <p><b>None:</b> Suggestions are not consolidated.</p>
<b>Delimiter</b>	<p>Specifies the delimiter to use between each suggestion. This is applicable if you chose <b>Delimited</b> for the <b>Style</b> option.</p> <p>Choose any character or string to separate each suggestion. This value should differ from the <b>Field Delimiter</b> value.</p>

Option/Option group	Description
<b>Field Delimiter</b>	<p>Specifies the delimiter to use between each suggestion list. This is applicable if you chose <b>Delimited</b> for the <b>Style</b> option.</p> <p>This value should differ from the <b>Delimiter</b> value.</p>
<b>Lastlines Match Minimum</b>	<p>Specifies the similarity score required for lastline suggestions.</p> <p>Enter a value from 0 to 80.</p> <p>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.</p>
<b>Match Range</b>	<p>Specifies whether to disregard an address-line suggestion when it does not match the primary range of the input address.</p> <p><b>Yes:</b> Returns address-line suggestions only when they match the primary range of the input address.</p> <p><b>No:</b> Returns a possible address-line suggestion when it doesn't have the same primary range as the input.</p>
<b>Max Number Addresslines</b>	<p>Specifies the maximum number of address-line suggestions that can be generated. The maximum number that you can enter is 100.</p> <p>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.</p> <p><b>Note:</b> If you set a low maximum, a viable suggestion could be left out of the suggestion list.</p>
<b>Max Number Lastlines</b>	<p>Specifies the maximum number of lastline suggestions that can be generated. The maximum number that you can enter is 15.</p> <p>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.</p> <p><b>Note:</b> If you set a low maximum, a viable suggestion could be left out of the suggestion list.</p>

Option/Option group	Description
<b>Style</b>	<p>Specifies the style of the output file.</p> <p><b>Delimited:</b> Outputs the suggestion list data in a delimited text format, with the delimiters specified in the <b>Delimiter</b> and <b>Field Delimiter</b> options.</p> <p><b>XML:</b> 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.</p>
<b>Suggestion List Components</b>	<p>Specifies the address field components that you want to include in the Suggestion_List output field.</p> <p><b>Note:</b> Suggestion list field components that do not have a value are not output to the Suggestion_List output field if the selected <b>Style</b> is <b>XML</b>.</p>

#### Related Topics

- [Suggestion List Components](#)
- [Designer Guide: Data Quality, Similarity score](#)
- [Designer Guide: Nested Data, Extracting data quality XML strings using extract\\_from\\_xml function](#)

### 5.4.13.13 Suggestion List Components

These options let you choose what information to output to the Suggestion\_List output field.

**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
<b>Firm</b>	Select Yes to output the firm name for the secondary address.
<b>Locality1</b>	Select Yes to output the locality1 (city) preferred by the USPS. Applicable for primary, secondary, and lastline address levels.

Option/option group	Description
<b>Postcode</b>	Select Yes to output the five-digit Postcode1 (not including the four-digit ZIP4). Applicable for primary, secondary, and lastline address levels.
<b>Postcode2 Even</b>	Select Yes to output the four-digit ZIP4 code, even numbers only. Applicable for primary and secondary address levels.
<b>Postcode2 Odd</b>	Select Yes to output the four-digit ZIP4 code, odd numbers only. Applicable for primary and secondary address levels.
<b>Primary Name1</b>	Select Yes to output the street name description. Applicable for primary and secondary address levels. <b>Note:</b> Primary numbers (high or low) are not output to this field.
<b>Primary Name Full1</b>	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. <b>Note:</b> Primary numbers (high or low) are not output to this field.
<b>Primary Number High</b>	Select Yes to output the high portion of the premise number range. Applicable for primary and secondary address levels.
<b>Primary Number Low</b>	Select Yes to output the low portion of the premise number range. Applicable for primary and secondary address levels.
<b>Primary Postfix1</b>	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.
<b>Primary Prefix1</b>	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.
<b>Primary Side Indicator</b>	Select Yes to output Odd or Even for the primary side indicator. Applicable for primary and secondary address levels.

Option/option group	Description
<b>Primary Type1</b>	Select Yes to output the abbreviated street type (for example, St, Ave, or Pl). Applicable for primary and secondary address levels.
<b>Region1</b>	Select Yes to output the state, province, territory, or region. Applicable for primary, secondary, and lastline address levels.
<b>Secondary Side Indicator</b>	Select Yes to output Odd or Even for the secondary side indicator. Applicable for secondary address level.
<b>Selection</b>	Select Yes to output the selection number for multiple suggestions.
<b>Unit Description</b>	Select Yes to output the unit description (for example, #, Apartment, or Flat). Applicable for secondary address level.
<b>Unit Number High</b>	Select Yes to output the high portion of the unit number range. Applicable for secondary address level.
<b>Unit Number Low</b>	Select Yes to output the low portion of the unit number range. Applicable for secondary address level.

### Related Topics

- [Output fields](#)

## 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
<b>Enable Z4 Change</b>	<p>Specifies whether to enable Z4Change processing.</p> <p><b>Yes:</b> Turns on Z4Change processing.</p> <p><b>No:</b> Turns off Z4Change processing.</p>
<b>Last ZIP4 Assign Date</b>	<p>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.</p> <p>Enter the date using the format MM/YYYY. For example, enter a date of January 2004 by typing 01/2004.</p> <p>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.</p>

### 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 Topics

- [Input fields](#)
- [Output fields](#)

#### 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	<p>Specifies the address class for the generated field.</p> <p><b>Delivery:</b> 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.</p> <p><b>Dual:</b> 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.</p> <p><b>Official:</b> When used with the applicable Field Name, this value generates fields in the form of the data preferred by the Postal Authority.</p> <p>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.</p>
Field Category	<p><b>Component:</b> Individual address components and postal codes that are related to the processed record.</p> <p><b>Standardized:</b> Standardized input lines based on the settings in the Standardization Options group in the transform.</p> <p><b>Suggestion:</b> Suggestion list output data based on the settings in the Suggestion List Options group.</p>

Category	Description
Field Class (USA Regulatory Address Cleanse)	<p>Specifies the field class that you want to assign to your output fields.</p> <p><b>Best:</b> 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.</p> <p><b>Note:</b> 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.</p> <p><b>Correct:</b> 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.</p> <p><b>Parsed:</b> Outputs the parsed value.</p> <p><b>Pre_LACSLink:</b> Retained address components that were replaced with LACSLink address information.</p> <p><b>Move_Updated:</b> Outputs the address components that have been updated with move-updated address data.</p> <p><b>Note:</b> 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.</p>
Field Class (Global Address Cleanse)	<p>Specifies the field class that you want to assign to your output fields.</p> <p><b>Best:</b> 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.</p> <p><b>Parsed:</b> Outputs the parsed value.</p>
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 Topics**

- [Content types](#)
- [Designer Guide: How address cleanse works](#)

**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").
Check_Digit	<p>The check-digit for 11-digit delivery-point bar code. Applicable only if the transform can make a full assignment.</p> <p>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).</p>
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	<p>The input source or list identifier.</p> <p>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.</p> <p>Statistics are generated for each unique value in this field when you map the field in conjunction with enabling the <b>Gather Statistics Per Data Source</b> option in the Reports and Statistics group.</p>
Delivery_Point	The two-digit DPBC code.
Family_Name1	The family name (for example, Smith).
Firm	The company name.

Field	Description
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	<p>The Line-of-Travel sortation:</p> <p><b>A:</b> Ascending</p> <p><b>D:</b> Descending</p> <p>LOT codes are required for non-automated, CART presorting in Standard Mail, Enhanced Carrier Route Subclass.</p>
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.).

Field	Description
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).

#### 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.  <b>Note:</b> If the output values don't fit within the length of the output field, then intelligent truncation occurs.

Field	Description
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.

Field	Description
Address_Type (DELIVERY, DUAL)	

Field	Description
	<p>The record-type indicator for the assigned address. Applicable for DELIVERY and DUAL Generated Field Address Class.</p> <p>The first character indicates the type of record in the address directory to which the address matched:</p> <p><b>F:</b> Firm</p> <p><b>G:</b> General delivery</p> <p><b>H:</b> High-rise apartment or office building</p> <p><b>M:</b> Military</p> <p><b>P:</b> Post office box</p> <p><b>R:</b> Rural route or highway contract</p> <p><b>S:</b> Street (usually, one side of one city block)</p> <p><i>blank:</i> Unassigned</p> <p>The second character may be a <b>D</b> 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.</p> <p><b>FD:</b> Firm default. The transform did not assign a firm-level Postcode2, but could do so if given more or better firm information.</p> <p><b>GD:</b> General delivery default. Assigned when General Delivery is the only primary name listed for the Postcode1.</p> <p><b>HD:</b> 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.</p> <p><b>RD:</b> 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.</p> <p><b>SD:</b> Street default. Usually means that there is no Postcode2 for the building, so the transform had to assign the Postcode2 for the block.</p> <p><b>UD:</b> Unique default. Either the owner of the unique Postcode1 has not provided Postcode2 assignments, or the address could not be matched.</p> <p>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</p>



Field	Description
	contract, general delivery, or PO Box was parsed.
AGeo_Countycode	Federal Information Processing Standard (FIPS) county code.
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.
AGeo_Placecode	Federal Information Processing Standard (FIPS) place code.
AGeo_Sectioncode	U.S. census tract code in the format 1234567890.
AGeo_Statecode	Federal Information Processing Standard (FIPS) state code.
Alias_Type (OFFICIAL)	<p>The alias-type indicator for the assigned address. Applicable for OFFICIAL Generated Field Address Class. It describes the input address, not the output address.</p> <p><b>A:</b> The input address matches an abbreviated street name.</p> <p><b>B:</b> The input address matches the high-rise alternate default base record.</p> <p><b>C:</b> The input street name is out of date; to get new street name, convert your record to the preferred alias.</p> <p><b>H:</b> The input address is an undesirable alternate, subject to conversion to a USPS preferred street address (high-rise alternate).</p> <p><b>O:</b> The input address is a street nickname or other alias.</p> <p><b>P:</b> The input address is a preferred alias.</p> <p><i>blank:</i> The input address is not an alias or is unassigned.</p>

Field	Description
ANKLink_Return_Code	<p>ANKLink return code (Attempted Not Known). Valid values are:</p> <p><b>77</b>: 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.</p> <p><i>blank</i>: No NCOALink lookup or no ANKLink match. This is always blank for full service providers.</p>
Audit_Dropped_Secondary	This field is used for audit testing. This field is also populated when an ANKLink match is made.
Audit_Prenome Audit_Given_Name1 Audit_Given_Name2 Audit_Family_Name1 Audit_Postname	<p>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).</p> <p>In the case of a firm match, these name fields contain a split version of the firm data.</p> <p>These fields are also populated when an ANKLink match is made.</p>
Audit_Gender	This field is used for audit testing. This field is also populated when an ANKLink match is made.
Audit_General	<p>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.</p> <p>For more information about the content of this field, see the NCOALink User Technical Reference at <a href="http://ribbs.usps.gov/">http://ribbs.usps.gov/</a>.</p> <p>This field can also contain ANKLink return codes.</p>
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.

Field	Description
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	<p>The carrier-route sort zone indicates eligibility for Standard Mail Automation Enhanced Carrier Route.</p> <p><b>A:</b> Carrier route rates are available and merging is allowed.</p> <p><b>B:</b> Carrier route rates are available and merging is not allowed.</p> <p><b>C:</b> Carrier route rates are not available and merging is allowed.</p> <p><b>D:</b> Carrier route rates are not available and merging is not allowed.</p>
CASS_Assignment_Type	<p>Indicates the option used in making the assignment:</p> <p><b>0:</b> The non-CASS and DPV tie-break options are disabled or not used to make an assignment.</p> <p><b>1:</b> Inexact Postcode1 move assignment.</p> <p><b>2:</b> Input Postcode2 assignment.</p> <p><b>3:</b> DPV tie-breaking was used to make this assignment.</p> <p><i>blank</i>: The transform cannot assign an input address.</p>

Field	Description
CASS_Record_Type	<p>The record type necessary for posting on the CASS test. This field is populated for assigned records only. The valid record types include:</p> <p><b>F:</b> Firm</p> <p><b>G:</b> General delivery</p> <p><b>H:</b> High-rise</p> <p><b>P:</b> Post office box</p> <p><b>R:</b> Rural route or highway contract</p> <p><b>S:</b> Street</p>
CGeo_BSACode	<p>A Core-Based Statistical Area (CBSA) that consists of:</p> <ul style="list-style-type: none"> <li>• A county with an incorporated place or a census-designated place that has a population of at least 10,000.</li> <li>• Adjacent counties with at least 25 percent of employed residents of the county who work in the CBSA's core or central county.</li> </ul> <p>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.</p>
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. The value 0000 indicates that the address does not lie in any MSA (usually a rural area).
CGeo_Sectioncode	<p>U.S. census tract and block group code. The first six digits are the tract number, and the first of the final four digits is the block group code within the tract.</p> <p>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.</p>

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	<p>Type of postal facility:</p> <p><b>A:</b> Airport Mail Facility (AMF)</p> <p><b>B:</b> Branch Office</p> <p><b>C:</b> Community Post Office (CPO)</p> <p><b>D:</b> Area Distribution Center (ADC)</p> <p><b>E:</b> Sectional Center Facility (SCF)</p> <p><b>F:</b> Delivery Distribution</p> <p><b>G:</b> General Mail Facility (GMF)</p> <p><b>K:</b> Network Distribution Centers (NDC)</p> <p><b>M:</b> Money Order Unit</p> <p><b>N:</b> City/place name</p> <p><b>P:</b> Post Office (main)</p> <p><b>S:</b> Station</p> <p><b>U:</b> Urbanization (Puerto Rico only)</p>

Field	Description
District	District number for the U.S. House of Representatives.
DPV_CMRA	<p>The DPV Commercial Mail Receiving Agencies (CMRA) component that is generated for this record.</p> <p><b>L:</b> The address triggered DPV locking.</p> <p><b>N:</b> The address is not a CMRA.</p> <p><b>Y:</b> The address is a valid CMRA.</p> <p><i>blank</i>: 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.</p>

Field	Description
DPV_Footnote	<p>DPV footnotes are required for end-user CASS certification. The footnotes contain the following information:</p> <p><b>AA:</b> The input address matches to the ZIP+4 file.</p> <p><b>A1:</b> The input address does not match to the ZIP+4 file.</p> <p><b>BB:</b> All input address field values match to DPV.</p> <p><b>CC:</b> The input address primary number matches to DPV, but the secondary number does not match (the secondary is present but invalid).</p> <p><b>F1:</b> The input address matches to a military address.</p> <p><b>G1:</b> The input address matches a general delivery address.</p> <p><b>M1:</b> The input address primary number is missing.</p> <p><b>M3:</b> The input address primary number is invalid.</p> <p><b>N1:</b> The input address primary number matches to DPV but the address is missing the secondary number.</p> <p><b>NL:</b> 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.</p> <p><b>Note:</b> The NL footnote is applicable only for the Move Updated generated field class.</p> <p><b>P1:</b> The input address is missing the rural route or highway contract box number.</p> <p><b>P3:</b> The input address is an invalid post office, rural route, or highway contract number.</p> <p><b>RR:</b> The input address matches to CMRA.</p> <p><b>R1:</b> The input address matches to CMRA, but the secondary number is not present.</p> <p><b>U1:</b> The input address matches a unique address.</p> <p><b>Note:</b> The transform always posts the DPV footers in the same order and this field is not always 12 characters in length.</p>

Field	Description
DPV_NoStats	<p>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.</p> <p><b>Y:</b> The address is flagged as No Stat in DPV data.</p> <p><b>N:</b> The address is not No Stat.</p> <p><i>blank:</i> The address was not looked up.</p> <p><b>Note:</b> The US Addressing report contains DPV NoStats counts in the DPV Summary section.</p>
DPV_Status	<p>The DPV status component that is generated for this record.</p> <p><b>D:</b> The primary range is a confirmed delivery point, but the secondary range is not available on input.</p> <p><b>L:</b> The address triggered DPV locking.</p> <p><b>N:</b> The address is not a valid delivery point.</p> <p><b>S:</b> The primary range is a valid delivery point, but the parsed secondary range is not valid in the DPV directory.</p> <p><b>Y:</b> The address is a confirmed delivery point. The primary range and secondary range (if present) are valid.</p> <p><i>blank:</i> 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.</p>
DPV_Vacant	<p>Vacant address indicator.</p> <p><b>Y:</b> The address is vacant.</p> <p><b>N:</b> The address is not vacant</p> <p><i>blank:</i> The address was not looked up.</p> <p><b>Note:</b> The US Addressing report contains DPV Vacant counts in the DPV Summary section.</p>



Field	Description
DSF2_Business_Indicator	<p>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.)</p> <p><b>Y:</b> Business address.</p> <p><b>N:</b> Not a business address.</p> <p><i>blank:</i> The address was not looked up.</p>
DSF2_Delivery_Type	<p>Delivery type.</p> <p><b>1:</b> Curb-side delivery.</p> <p><b>2:</b> NDCBU (Neighborhood Delivery Centralized Box Unit) delivery.</p> <p><b>3:</b> Central delivery.</p> <p><b>4:</b> Door-slot delivery.</p> <p><i>blank:</i> The address was not looked up.</p>
DSF2_Drop_Count	<p>Drop count.</p> <p>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.</p>
DSF2_Drop_Indicator	<p>Drop indicator.</p> <p><b>Y:</b> The delivery point serves multiple businesses or families. For example, delivery point may be a CMRA (Commercial Mail Receiving Agency).</p> <p><b>N:</b> The delivery address is not a CMRA.</p> <p><i>blank:</i> The address was not looked up.</p>
DSF2_Educational_Ind	<p>Educational indicator.</p> <p><b>Y:</b> The address is an educational institution.</p> <p><b>N:</b> The address is not an educational institution.</p> <p><i>blank:</i> The address was not looked up.</p>

Field	Description
DSF2_LACS_Conversion_Indicator	LACS (Locatable Address Conversion System) indicator. <b>Y</b> : The address is LACS convertible. <b>N</b> : The address is not LACS convertible. <i>blank</i> : The address was not looked up.
DSF2_Record_Type	Record type. <b>B</b> : Business address. <b>R</b> : Residential address. <b>U</b> : Unknown. AP.DSF_Deltype is blank. <i>blank</i> : No information available.
DSF2_Seasonal_Indicator	Seasonal address indicator. <b>Y</b> : The address is seasonally occupied. <b>N</b> : The address is not seasonal. <i>blank</i> : The address was not looked up.
DSF2_Throwback_Indicator	Throwback indicator. <b>Y</b> : Customer with street address wants delivery at PO Box instead. <b>N</b> : No throwback necessary. <i>blank</i> : The address was not looked up.

Field	Description
Error	<p>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.</p> <p><b>0:</b> There are no suggestion selection errors.</p> <p><b>1:</b> 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.</p> <p><b>2:</b> The suggestion selection is invalid. For example, 8 was selected but there are only five suggestions.</p> <p><b>3:</b> The suggestion entry in the input field is invalid.</p> <p><b>4:</b> The suggestion range in the input field is invalid.</p> <p><b>5:</b> The suggestion secondary range in the input field is invalid.</p>
EWS_Match	<p>Returns the results of the EWS (Early Warning System) match.</p> <p><b>T:</b> True, the address is located in the EWS directory and is an EWS match.</p> <p><b>F:</b> False, the address is not located in the EWS directory.</p> <p><i>blank:</i> EWS is not enabled.</p>
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.
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.

Field	Description
Fault_Code	The fault code. This is blank if the address is assigned. For more information, see <a href="#">Fault codes (USA Regulatory Address Cleanse)</a> .
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.  <b>F:</b> Foreign addresses  <i>blank:</i> Domestic U.S.
Full_Address	The complete address line, including secondary address and dual address (street and postal) line data.  <b>Note:</b> This field may contain invalid secondary address information when you set the <b>Include Unused Address Line Data</b> option in the Standardization Options group to <b>Yes</b> .  <b>Note:</b> If the output values don't fit within the length of the output field, then intelligent truncation occurs.

Field	Description
Geo_Matchcode	<p>Match code indicating the precision of the latitude and longitude assignment.</p> <p><b>0:</b> Matches in address level.</p> <p><b>1:</b> Nine-digit match. Usually indicates precision to a particular block face.</p> <p><b>4:</b> Seven-digit match. Usually indicates precision within a few blocks.</p> <p><b>5:</b> Five-digit match. Usually indicates precision to within a mile or two.</p> <p><b>7:</b> No match in centroid-level.</p> <p><b>8:</b> No match in address-level.</p> <p><b>9:</b> No match in centroid-level or address-level.</p> <p><i>blank:</i> No centroid-level or address-level directory lookup performed.</p>
Intermediate_Codes	<p>Intermediate codes provide information that the USPS requires when you perform NCOALink certification or audit testing.</p>
LACSCode	<p>LACS (Locatable Address Conversion System) indicator.</p> <p><b>T:</b> The address needs 9-1-1 conversion (from box to street address) and should be submitted to a LACS vendor.</p> <p><b>F:</b> The address does not need conversion.</p> <p><i>blank:</i> The address was not assigned.</p>
LACSLink_Indicator	<p>Returns the conversion status of addresses processed by LACSLink.</p> <p><b>Y:</b> The address was converted by LACSLink (the LACSLink_Return_Code value is A).</p> <p><b>N:</b> The address was looked up with LACSLink but not converted.</p> <p><b>F:</b> The address was a false-positive.</p> <p><b>S:</b> A LACSLink conversion was made, but it was necessary to drop the secondary information.</p> <p><i>blank:</i> No LACSLink lookup attempted.</p>

Field	Description
LACSLink_Query	<p>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 (#).</p> <p><i>blank</i>: No LACSLink lookup attempted.</p>
LACSLink_Return_Code	<p>Returns the match status for LACSLink processing.</p> <p><b>A</b>: LACSLink record match. A converted address is provided in the address data fields.</p> <p><b>00</b>: No match and no converted address.</p> <p><b>09</b>: LACSLink matched an input address to an old address, which is a "high-rise default" address; no new address is provided.</p> <p><b>14</b>: Found a LACSLink record, but couldn't convert the data to a deliverable address.</p> <p><b>92</b>: LACSLink record matched after dropping the secondary number from input address.</p> <p><i>blank</i>: No LACSLink lookup attempted.</p>
Lastline	<p>Locality, region, and postal code together on one line.</p> <p><b>Note:</b> If the output values don't fit within the length of the output field, then intelligent truncation occurs.</p>
Locality1	<p><b>Canada and USA engines:</b> Locality preferred by the postal authority.</p> <p><b>Other engines:</b> City, town, locality, or suburb.</p> <p><b>Note:</b> If the output values don't fit within the length of the output field, then intelligent truncation occurs.</p>

Field	Description
Locality1_Alternate	<p>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.</p> <p>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.</p>
Locality1_LLIDX	<p>Yields a city name (locality1 name) that is more geographically precise than Locality1_Official.</p> <p>LLIDX (last-line index) is a USPS number that ties a ZIP+4 record to a particular city, state, and ZIP.</p> <p><b>Note:</b> If the output values don't fit within the length of the output field, then intelligent truncation occurs.</p>
Locality1_Name	<p>The city, town, or suburb.</p> <p>Locality names that are marked as invalid for mailing by the USPS are always preserved, never converted, regardless of the values set for the <b>Preserve Place Name</b> and <b>Assign With Input Locality</b> options.</p>
Locality1_Official	<p>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 <b>Preserve Place Name</b> option is set.</p>
Locality1_Official_ABBR	<p>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.</p>
Locality2	<p><b>USA engine:</b> Urbanization (Puerto Rican addresses only).</p> <p><b>Other engines:</b> Additional city, town, locality, or suburb information.</p>
Locality2_Official	<p>Urbanization name; produced only when the address is in Puerto Rico.</p>
LOT	<p>Line-of-travel number.</p>

Field	Description
LOT_Order	Line-of-travel sortation. <b>A:</b> Ascending. <b>D:</b> Descending.
Matched_Addressline_Indicator	Match level indicator. <b>T:</b> The address line is matched to a ZIP+4 record. <b>F:</b> The address line is not matched to a ZIP+4 record.
Matched_Lastline_Indicator	Match level indicator. <b>T:</b> The last line is matched to a City/ZCF record. <b>F:</b> 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 yyyyymm. The yyyyymm 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>B:</b> Business (matched by company name). <b>F:</b> Family (matched by last name). <b>I:</b> Individual (matched by first and last name).  This field is also populated when an ANKLink match is made.



Field	Description
Multiline1-12	<p>A line that may contain any data. The type of data in this line may vary from record to record.</p> <p><b>Note:</b> These fields may contain invalid secondary address information when you set the <b>Include Unused Address Line Data</b> option in the "Standardization Options" group to <b>Yes</b>.</p> <p><b>Note:</b> For address data, if the output values don't fit within the length of the output field, then intelligent truncation occurs.</p>
Name	The name of a person associated with the address.
NCOALink_Hint_Byte	This field is used for audit testing.
NCOALink_Return_Code	<p>This field shows NCOALink return codes. To populate this field, set the <b>List Processing Mode</b> to one of the three available options: <b>Change of Address</b>, <b>Statistics Only</b>, or <b>Return Codes Only</b>.</p> <p>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 <b>Generate Return Code Descriptions</b> option in the NCOALink Report Options group.</p> <p>This field is also populated when an ANKLink match is made.</p>
Stage_Test_Record	<p>This field is for stage testing only. It applies to NCOALink, CASS, DSF2 Augment, DSF2 Sequence, and DSF2 Invoice self-certifications.</p> <p>The USA Regulatory Address Cleanse transform populates the values of this field automatically to match the format required for stage testing.</p>
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	<p>The secondary address match that is made using the input ZIP+4 for missing or invalid secondary address information.</p> <p><b>Note:</b> If the output values don't fit within the length of the output field, then intelligent truncation occurs.</p>

Field	Description
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.
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. <b>M:</b> Military. <b>U:</b> Unique (specific to a university, large firm, or other institution). <i>blank</i> : Ordinary ZIP Code or the ZIP Code was not assigned.
Postcode1	The five-digit ZIP Code. Does not include the four-digit ZIP4.

Field	Description
Postcode1_Change_Ind	<p>Indicates whether the address is affected by postal code realignment.</p> <p><b>T:</b> True, the transform corrected the postal code (and the locality, if applicable).</p> <p><b>F:</b> False.</p> <p><i>blank:</i> The address was not corrected.</p>
Postcode2	<p>Four-digit ZIP4 Code. On a mail piece, this code follows the primary postal code, with a hyphen placed between, for example, 54601-1234.</p>
Pre_Suitelink_Delivery_Point	<p>The numeric two-digit code for the delivery point bar code that was generated before SuiteLink processing.</p>
Pre_Suitelink_Postcode1	<p>The ZIP Code that was assigned by the transform before SuiteLink processing.</p> <p><b>5-digit ZIP Code:</b> SuiteLink Retcode value is A.</p> <p><i>blank:</i> No ZIP Code assigned.</p>
Pre_Suitelink_Postcode2	<p>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.</p>
Pre_Suitelink_Unit_Description	<p>The unit designator that existed before SuiteLink processing. If this field is blank, the transform did not assign any secondary information.</p>
Pre_Suitelink_Unit_Number	<p>The secondary range information that existed before SuiteLink processing. If this field is blank, the transform did not assign any secondary information.</p>
Primary_Address	<p>Primary address line, such as the street address or post office box. Does not include secondary address information such as apartment.</p> <p>If the <b>Use USPS Primary Name Abbreviation</b> 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.</p>
Primary_Name1	<p>Street name description.</p> <p><b>Note:</b></p> <p>If the output values don't fit within the length of the output field, then intelligent truncation occurs.</p>

Field	Description
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	<p>The primary address and secondary address on one line. Does not include remainder data. This line is always output as if the <b>Include Unused Address Line Data</b> option is set to <b>No</b>, which means that the output does not include invalid secondary address line information.</p> <p><b>Note:</b> If the output values don't fit within the length of the output field, then intelligent truncation occurs.</p>
Primary_Type1	Abbreviated street type (for example, St, Ave, or Pl).
Primary_Type1_Long	Fully-spelled street type (for example, Street or Avenue).
QSS_Default	<p>Specifies whether the record qualified as a default match instead of qualifying as a match at a higher level of assignment. Output values are:</p> <p><b>T:</b> True</p> <p><b>F:</b> False</p>
RDI_Indicator	<p>The residential delivery indicator (RDI) shows whether the address is residential or nonresidential.</p> <p><b>Y:</b> Residential address</p> <p><b>N:</b> Nonresidential address</p>

Field	Description
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 five-digit ZIP Code or two-digit zone.
Sortcode_Route	The four-digit carrier route.
Status	<p>Specifies the suggestion status generated as the result of looking up the current record and performing suggestion processing.</p> <p><b>A:</b> Suggestion processing ended with an address suggestion list needing resolution.</p> <p><b>L:</b> Suggestion processing ended with a lastline suggestion list needing resolution.</p> <p><b>N:</b> No suggestion lists were generated and no suggestion processing was performed.</p> <p><b>R:</b> The primary range is invalid for the selected address suggestion.</p> <p><b>S:</b> The secondary range is invalid for the selected address suggestion.</p> <p><b>U:</b> The secondary address is invalid for the selected address suggestion.</p>
Status_Code	<p>The status code. This field is blank if the address is unassigned.</p> <p>For more information, see <a href="#">Status codes (USA Regulatory Address Cleanse)</a>.</p>
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	<p><b>A:</b> SuiteLink match—Secondary information exists and was assigned to this record as a result of SuiteLink processing.</p> <p><b>00:</b> No SuiteLink match—Lookup was attempted but no matching record was found.</p> <p><i>blank:</i> A SuiteLink lookup was not attempted because one of the following is true:</p> <ul style="list-style-type: none"> <li>• The address is not a high-rise default according to CASS.</li> <li>• The address does not contain a firm.</li> </ul>
Undeliverable_Indicator	<p>Indicates whether the record is a deliverable address.</p> <p><b>T:</b> The address is tagged by the USPS as unsuitable for mail delivery (for example, a cemetery).</p> <p><b>F:</b> 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.</p>
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 Topics

- [Fault codes \(USA Regulatory Address Cleanse\)](#)
- [Status codes \(USA Regulatory Address Cleanse\)](#)

## 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 Topics

- [Designer Guide: Data Flows, Associate, Match, and User-Defined transform editors](#)
- [Designer Guide: Data Flows, Data Quality transform editors](#)

## 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 Topics

- [Transform configurations](#)
- [Downloading blueprints and other content objects](#)

## 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	<p>Specifies how the Python expression is applied to the transform.</p> <p><b>Per collection:</b> 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.</p> <p><b>Per record:</b> 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.</p>

### Related Topics

- [Group forming](#)
- [User-Defined Transform options](#)



#### 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.

##### **Note:**

Candidate selection works with relational databases only; it does not work with flat files.

##### **Related Topics**

- [Break group options](#)
- [Candidate selection options](#)

#### *Break group options*

Use the break group options to group records based on common field values.

Options	Description
<b>Split records into break groups</b>	<p>Select this option if you want to form break groups to reduce the total number of comparisons made.</p> <p>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.</p> <p><b>Caution:</b> 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 <b>Maximum allowable break group size (in records)</b> option so that the collection does not exceed the size limit. If it does exceed the limit, the data flow will abort.</p> <p><b>Note:</b> Break group size is calculated by multiplying the record length by the number of records in the break group.</p>
<b>Field</b>	<p>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.</p> <p>If you require a more complex break key, you could define that field using an upstream Query transform and select the field here.</p>
<b>Start Position</b>	<p>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.</p> <p>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).</p>
<b>Length</b>	Enter the number of characters in the field you want included in the break key.
<b>Break key case sensitive</b>	<p>Specifies whether to treat the break key as case sensitive.</p> <p><b>Yes:</b> Treat the break key as case sensitive.</p> <p><b>No:</b> Do not treat the break key as case sensitive.</p> <p>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.</p>

Options	Description
<b>Replace NULL with empty string</b>	<p>Specifies whether to convert NULL values with an empty string in the break key.</p> <p><b>Yes:</b> Convert NULL to an empty string.</p> <p><b>No:</b> Do not convert to an empty string.</p>
<b>Right pad fields with blanks</b>	<p>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.</p> <p>If the <b>Replace NULL with empty string</b> 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).</p> <p><b>Yes:</b> Right-pad fields with blank spaces.</p> <p><b>No:</b> Do not right-pad fields.</p>
<b>Input already sorted</b>	<p>Specifies that the input data has already been sorted, and you do not want it sorted again.</p> <p>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.</p> <p><b>Yes:</b> The transform will not re-sort the input data.</p> <p><b>No:</b> The transform will sort the break keys at runtime before forming break groups.</p>
<b>Maximum allowable break group size (in records)</b>	<p>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.</p> <p>With this option, you can control the amount of memory used during processing by specifying the number of records processed at one time.</p> <p>If more records make it into a single break group than specified, then the dataflow throws an error and stops.</p>

### Related Topics

- [Designer Guide: Match, Break keys and candidate selection](#)

*Candidate selection options*

The candidate selection option group includes the following options:

Option	Description
<b>Datastore</b>	<p>Select a valid datastore.</p> <p>This list is populated with all valid SQL and persistent cache datastores.</p> <p>If you choose a persistent cache datastore, you will not be able to enter custom SQL.</p>
<b>Cache type</b>	<p>This option can be used to improve performance, with a trade-off of more memory consumption.</p> <p><b>No_Cache:</b> Specifies that each query will be sent to the database.</p> <p><b>Pre_Load_Cache:</b> Specifies that the entire secondary table is cached to a local disk or memory.</p>
<b>Auto-generate SQL</b>	<p>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 <b>Create custom SQL</b> option.</p>
<b>Table</b>	Enter a valid table name from the datastore.
<b>Use break column from database</b>	Select this option if your database already contains a column that corresponds to the break key field.
<b>Break key field</b>	Select the column from the secondary table that contains the break key field.
<b>Create custom SQL</b>	Select to create custom SQL.
<b>Launch SQL Editor</b>	Opens the SQL editor. This button is only enabled if you select the <b>Create custom SQL</b> option.
<b>Use constant source value</b>	Select to assign records to a physical source for generating appropriate statistics.
<b>Physical source value</b>	Type a value for your physical source. This value will be placed in the physical source field you select.
<b>Physical source field</b>	Select the mapped field that contains the physical source name.

Option	Description
<b>Add DB columns to mapping table</b>	<p>If you are using the <b>Create custom SQL</b> 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.</p> <p>If you are using the <b>Auto-generate SQL</b> option, clicking this button will add ALL database columns, in the order that they appear in the table schema.</p> <p><b>Note:</b> 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.</p>

### Column mapping table

This table allows you to specify which mapped field in the dataflow each database selected field is assigned to.

Column	Description
<b>Break key</b>	Specifies whether this field is used as part of your break key.
<b>Field</b>	Each cell contains a list of the mapped names from the input fields in the transform.
<b>DB column</b>	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 Topics

- [Designer Guide: Match, Break keys and candidate selection](#)

## 5.4.14.2 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.

Option	Description
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	<p>This option creates a separate sub data flow process for the transform when Data Services executes the data flow.</p> <p><b>Yes:</b> Splits transform into separate process.</p> <p><b>No:</b> Keeps transform in same process as the rest of the data flow.</p>

#### *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 Topics**

- [Python](#)
- [Create an expression with the Python Expression editor](#)

## **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).

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

### 5.4.15.1 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 5-199: 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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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



Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
Bermuda	BM	BMU	060		G	C, L
Bhutan	BT	BTN	064		G	C, L
Bolivia	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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
Central African Republic	CF	CAF	140		G	C, L
Chad	TD	TCD	148		G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
Cote d'Ivoire	CI	CIV	384		G	C, L
Croatia (Hrvatska)	HR	HRV	191	HR	G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Cuba	CU	CUB	192		G	C, L
Curaçao	CW	CUW	531		G	C, L
Cyprus	CY	CYP	196	CY	G	C, L
Czech Re- public (Czechoslo- vakia)	CZ	CZE	203	CZ	G	C, L, Pn, Pr
Democratic People's Re- public of Ko- rea	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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
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
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



Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
Libyan Arab Jamahiriya	LY	LBY	434		G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
Liechtenstein (Included in the Switzerland data package)	LI	LIE	438	FL	G	C, L, Pn, Pr
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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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 France data package)	MC	MCO	492	F	G	C, L, Pn, Pr
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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
Occupied Palestinian Territory	PS	PSE	275		G	C

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
Russian Federation	RU	RUS	643	RUS	G	C, L
Rwanda	RW	RWA	646		G	C, L
Saint Barthelemy (Included in the France data package)	BL	BLM	652		G	C, L
Saint Helena	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 (Included in the France data package)	MF	MAF	663		G	C, L

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
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	SX	SXM	534		G	C, L
Slovakia	SK	SVK	703		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
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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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
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

Country name	2-char ISO code	3-char ISO code	3-digit ISO code	European Postcode prefix	Engine	Assignment level
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

#### 5.4.15.2 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

Information code	Description	Engine(s)
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
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

Information code	Description	Engine(s)
3250	The range or building name is missing on input or both are not in the directory.	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
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
6000	Unclassified error.	All engines

### 5.4.15.3 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.
6	Corrected postal code and region.
7	Corrected country, postal code, and region.

Value	Description
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.



Value	Description
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.
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).

Value	Description
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.
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.

Value	Description
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.

### 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	Archived record used for assignment. Global Address engine.

Value	Description
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).

#### 5.4.15.4 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.

Quality code	Description
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.

#### 5.4.15.5 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.

Fault Code	Description
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.
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.

#### 5.4.15.6 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	<p><b>A:</b> The transform truncated the address line to make it fit your field.</p> <p><b>B:</b> The transform truncated both the address line and the Locality1_Name.</p> <p><b>C:</b> The transform truncated the Locality1_Name to make it fit your field.</p> <p><b>S:</b> No truncation occurred.</p>
2nd	<p><b>0:</b> Regarding the Locality1_Name, Region1, Postcode1, and Postcode2, there is no significant difference between the input data and the data that the transform assigned.</p> <p><b>1:</b> The transform assigned a different Postcode1.</p> <p><b>2:</b> The transform assigned a different Locality1_Name.</p> <p><b>3:</b> The transform assigned a different Locality1_Name and Postcode1.</p> <p><b>4:</b> The transform assigned a different Region1.</p> <p><b>5:</b> The transform assigned a different Region1 and Postcode1.</p> <p><b>6:</b> The transform assigned a different Locality1_Name and Region1.</p> <p><b>7:</b> The transform assigned a different Locality1_Name, Region1, and Postcode1.</p> <p><b>8:</b> The transform assigned a different Postcode2.</p> <p><b>9:</b> The transform assigned a different Postcode1 and Postcode2.</p> <p><b>A:</b> The transform assigned a different Locality1_Name and Postcode2.</p> <p><b>B:</b> The transform assigned a different Locality1_Name, Postcode1, and Postcode2.</p> <p><b>C:</b> The transform assigned a different Region1 and Postcode2.</p> <p><b>D:</b> The transform assigned a different Region1, Postcode1, and Postcode2.</p> <p><b>E:</b> The transform assigned a different Locality1_Name, Region1, and Postcode2.</p> <p><b>F:</b> The transform assigned a different Locality1_Name, Region1, Postcode1, and Postcode2.</p>

Digit	Description
3rd	<p><b>0:</b> Regarding the primary name, primary prefix/postfix, and primary type, there is no significant difference between the input and what the transform assigned.</p> <p><b>1:</b> The transform assigned a different primary type.</p> <p><b>2:</b> The transform assigned a different primary prefix.</p> <p><b>3:</b> The transform assigned a different primary prefix and primary type.</p> <p><b>4:</b> The transform assigned a different primary postfix.</p> <p><b>5:</b> The transform assigned a different primary type and primary postfix.</p> <p><b>6:</b> The transform assigned a different primary prefix and primary postfix.</p> <p><b>7:</b> The transform assigned a different primary prefix, primary type, and primary postfix.</p> <p><b>8:</b> The transform assigned a different primary name.</p> <p><b>9:</b> The transform assigned a different primary name and primary type.</p> <p><b>A:</b> The transform assigned a different primary prefix and primary name.</p> <p><b>B:</b> The transform assigned a different primary prefix, primary name, and primary type.</p> <p><b>C:</b> The transform assigned a different primary name and primary postfix.</p> <p><b>D:</b> The transform assigned a different primary name, primary type, and primary postfix.</p> <p><b>E:</b> The transform assigned a different primary prefix, primary name, and primary postfix.</p> <p><b>F:</b> The transform assigned a different primary prefix, primary name, primary postfix, and primary type.</p>



Digit	Description
4th	<p><b>0:</b> 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.</p> <p><b>1:</b> The transform assigned a different unit description.</p> <p><b>2:</b> The transform assigned a different delivery point.</p> <p><b>3:</b> The transform assigned a different delivery point and unit description.</p> <p><b>4:</b> The transform assigned a different sort code route.</p> <p><b>5:</b> The transform assigned a different sort code route and unit description.</p> <p><b>6:</b> The transform assigned a different sort code route and delivery point.</p> <p><b>7:</b> The transform assigned a different sort code route, delivery point, and unit description.</p> <p><b>8:</b> The transform assigned a different county number.</p> <p><b>9:</b> The transform assigned a different county number and unit description.</p> <p><b>A:</b> The transform assigned a different county number and delivery point.</p> <p><b>B:</b> The transform assigned a different county number, delivery point, and unit description.</p> <p><b>C:</b> The transform assigned a different county number and sort code route.</p> <p><b>D:</b> The transform assigned a different county number, sort code route, and unit description.</p> <p><b>E:</b> The transform assigned a different county number, sort code route, and delivery point.</p> <p><b>F:</b> The transform assigned a different county number, sort code route, delivery point, and unit description.</p>
5th	<p><b>0:</b> Regarding the LOT, LOT_Order, and Locality2_Official, there is no significant difference between the input data and the data that the transform assigned.</p> <p><b>1:</b> The transform assigned a different LOT.</p> <p><b>2:</b> The transform assigned a different LOT_Order.</p> <p><b>3:</b> The transform assigned a different LOT and LOT_Order.</p> <p><b>4:</b> The transform assigned a different Locality2_Official.</p> <p><b>5:</b> The transform assigned a different Locality2_Official and LOT.</p> <p><b>6:</b> The transform assigned a different Locality2_Official and LOT_Order.</p> <p><b>7:</b> The transform assigned a different Locality2_Official, LOT, and LOT_Order.</p>

Digit	Description
6th	Always outputs a zero (0).

### 5.4.15.7 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 Topics**

- [USA ShowA command line options](#)

#### 5.4.15.7.1 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.

#### 5.4.15.7.2 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.
-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 <code>dpbc</code> .
-4:zip4	/4:zip4	Enter the postcode2 for <code>zip4</code> .
-ad:file	/ad:file	Enter the Address-line dictionary and path name ( <code>addrln.dct</code> ) for <code>file</code> .
-c:cart	/c:cart	Enter the carrier route number for <code>cart</code> .
-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 <code>file</code> .
-nd:file	/nd:file	Enter the National ZIP+4 directory path and name ( <code>zip4us.dir</code> ) for <code>file</code> .
-pre:dir	/pre:dir	Enter the primary prefix (N, NE, E, SE, S, SW, W, NW) for <code>dir</code> .
-pos:dir	/pos:dir	Enter the primary postfix (N, NE, E, SE, S, SW, W, NW) for <code>dir</code> .
-s:street	/s:street	Enter the street primary name (in quotes if multiple words) for <code>street</code> .
-sfx:suffix	/sfx:suffix	Enter the primary type (Ave, Blvd, St, Rd, and so on) for <code>suffix</code> .

UNIX	Windows	Description
-sh:range	/sh:range	Enter the street (primary) range high for <code>range</code> .
-sl:range	/sl:range	Enter the street (primary) range low or exact for <code>range</code> .
-t:type	/t:type	Enter the file type (dBASE3, ASCII, or DELIMITED) for <code>type</code> .
-u:urb idx	/u:urb idx	Enter the urbanization Index for <code>urb idx</code> .
-z:lo-hi	/z:lo-hi	Enter the low and high range for postcode1 for <code>lo-hi</code> .
-z:zip	/z:zip	Enter the postcode1 for <code>zip</code> .

### Related Topics

- [ShowA/ShowL modes of operation](#)

#### 5.4.15.7.3 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.

UNIX	Windows	Description
-ab:query	/ab:query	Enter the abbreviated locality1 for <code>query</code> .
-cd:file	/cd:file	Enter the City directory path and name ( <code>city04.dir</code> ) for <code>file</code> .
-cn:city	/cn:city	Enter locality1 name (in quotes if multiple words) for <code>city</code> .
-dr:dir	/dr:dir	Enter which directory to search, City or ZCF for <code>dir</code> .
-f:file	/f:file	Enter the output file path and name for <code>file</code> .
-ml:query	/ml:query	Enter the military postcode1 for <code>query</code> .
-mz:query	/mz:query	Enter the multi-zone locality1 for <code>query</code> .
-pn:query	/pn:query	Enter a place name for <code>query</code> .
-st:state	/st:state	Enter region1 (for US use USPS abbreviations or full state names) for <code>state</code> .
-t:type	/t:type	Enter the output file type (dBASE3, ASCII, or DELIMITED) for <code>type</code> .
-un:query	/un:query	Enter the unique postcode1 for <code>query</code> .
-z:zip	/z:zip	Enter the postcode1 for <code>zip</code> .
-zd:file	/zd:file	Enter the ZCF directory path and name ( <code>zcf04.dir</code> ) for <code>file</code> .
-z:lo-hi	/z:lo-hi	Enter the postcode1 range for <code>lo-hi</code> .

### Related Topics

- [ShowA/ShowL modes of operation](#)

## 5.5 Platform transforms

### 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.

#### 5.5.1.1 Data inputs

Only one data flow source is allowed.

#### 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.

### 5.5.1.3 Options

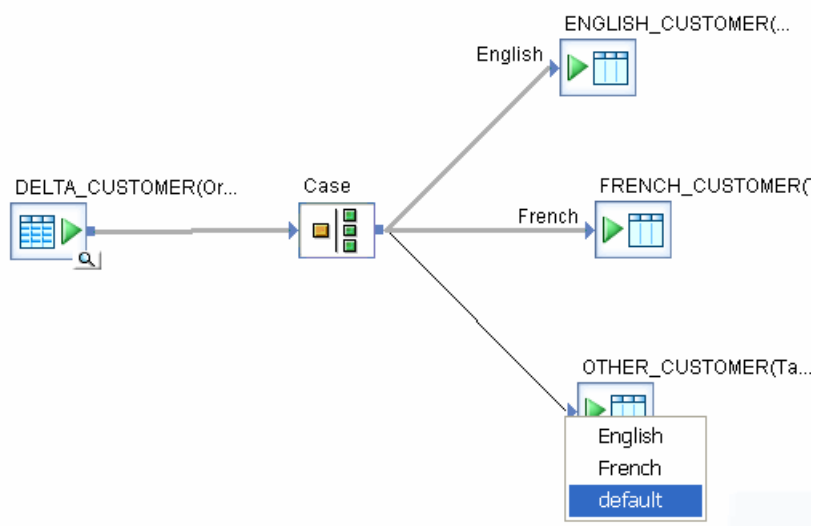
Option	Description
<b>Expression</b>	<p>CASE expression for the corresponding label.</p> <p>DEFAULT is the expression used when all other CASE expressions evaluate to false. To enable DEFAULT, select the <b>Produce default output when all expressions are false</b> check box. The label changes to read <b>Produce default output with label: default</b>. Change the label name if desired.</p>
<b>Label</b>	Name of the connection description indicating where data will go if the corresponding CASE condition is true.
<b>Preserve expression order</b>	<p>This option is available only when the <b>Row can be TRUE for one case only</b> option is checked.</p> <p>Select this option if expression order is important to you.</p> <p>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.</p> <p>This reordering of expressions can change your results, because there is no way to guarantee which expression will evaluate to TRUE first.</p>



Option	Description
<b>Produce default output with label</b> or <b>Produce default output when all expressions are false</b>	This option changes depending on whether it is selected. Select <b>Produce default output with label</b> to send rows to this case when all other case conditions are false.
<b>Row can be TRUE for one case only</b>	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 6.1 or later, this option defaults to TRUE

### 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

## 5.5.2 Map\_Operation



Allows conversions between data manipulation operations.

The Map\_Operation transform allows you to 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 could be to convert UPDATE rows to INSERT rows to preserve the existing row in the target.

Data Services can push Map\_Operation transforms to the source database.

### Related Topics

- [Effective\\_Date](#)
- [real](#)

### 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.

### 5.5.2.2 Options

Option	Description
<b>Output row type</b>	Indicate the new operations desired for the input data set. Choose from the following operation codes: INSERT, UPDATE, DELETE, NORMAL, or DISCARD.

### 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

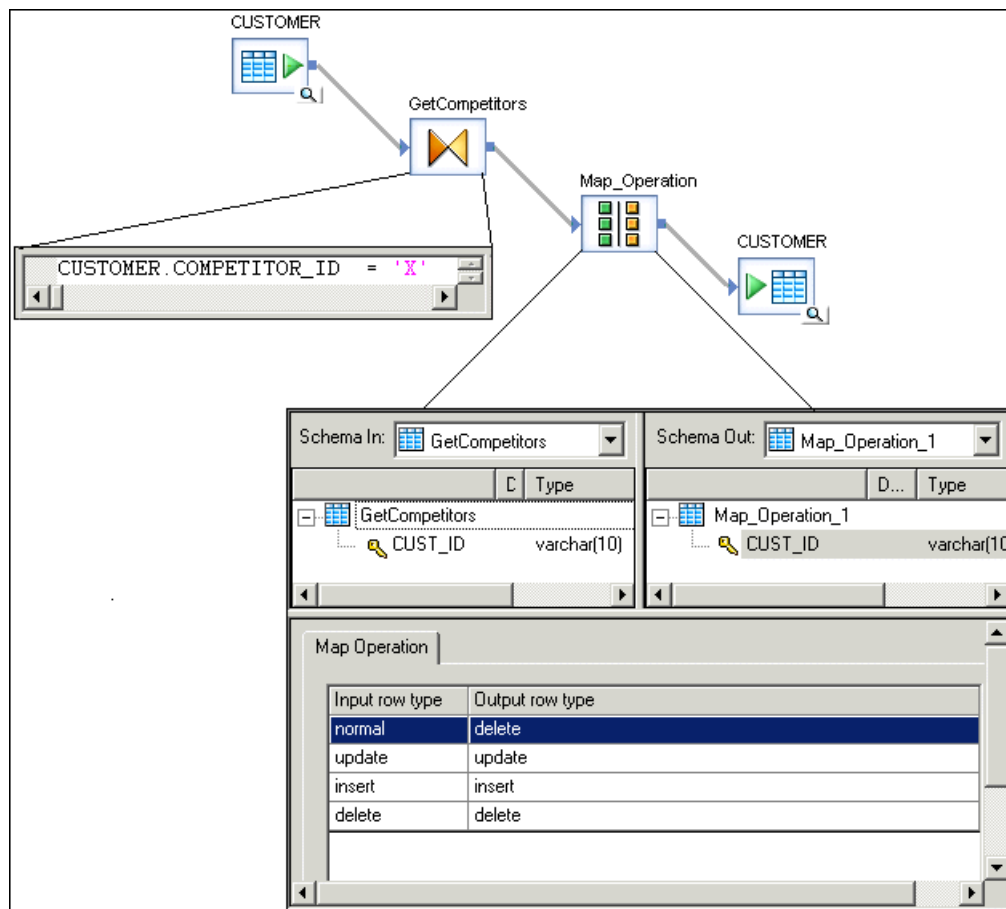
Any of these operation codes can be mapped to:

- 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.

In the following example, the Map\_Operation is used to delete all customers who have also been indicated as competitors:



### 5.5.3 Merge



Combines incoming data sets, producing a single output data set with the same schema as the input data sets.

#### 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.

### 5.5.3.2 Options

None.

### 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.

## 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 Topics**

- [Query transform output schema](#)
- [Query transform input schema](#)
- [Smart editor](#)

### 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.

**Note:**

Use caution when using columns of data type `real` in this transform. Comparison results are unpredictable for the `real` data type.

### 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.

### 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.

### 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.

#### 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.







Command	Applicable elements	Effect
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.

#### 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 <b>Validation &gt; Validate</b> .

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 Topics

- [Change the current schema](#)

### 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 BusinessObjects 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.

Command	Applicable elements	Effect
Paste	All	<p>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.</p> <p>If the cursor overlaps an existing column, you are prompted to insert above, insert below, remap column, or cancel.</p> <p><b>Note:</b> 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.</p>
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	<p>Adds a function or stored procedure call to the current schema. The function or procedure must already be imported into the repository.</p> <p>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).</p> <p>In the Define Input Parameter(s) window, map all first-level input parameters for the function to the input parameters of the query.</p>


Command	Applicable elements	Effect
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 up-stream 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 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.

Command	Applicable elements	Effect
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 Topics

- [Query transform output schema](#)

## 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  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 5-218: 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.

Tab	Description
WHERE	<p>Specifies conditions that determine which rows are output.</p> <p>Enter the conditions in SQL syntax, like a WHERE clause in a SQL SELECT statement. For example:</p> <pre>TABLE1.EMPNO = TABLE2.EMPNO AND TABLE1.EMPNO &gt; 1000 OR TABLE2.EMPNO &lt; 9000</pre> <p>Use the <b>Functions</b>, <b>Domains</b>, and <b>smart editor</b> buttons for help building expressions.</p>
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 Topics**

- [Mapping tab](#)
- [SELECT tab](#)
- [FROM tab](#)
- [WHERE tab](#)
- [GROUP BY tab](#)
- [ORDER BY tab](#)
- [Advanced tab](#)
- [Find tab](#)

**5.5.4.9 Query Editor tabs**

#### 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.

#### *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.

#### 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.


#### 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	<p>The left source of a join.</p> <p>For the first join pair, select the input schema from the drop-down list of available schemas.</p> <p>For subsequent join pairs, the result of the previous join pair is taken as the left source and the schema cannot be specified.</p>
Join Type	<p>The type of join.</p> <p>Valid values are Inner join and Left outer join.</p> <p>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.</p>
Right	<p>The right source of a join.</p> <p>Any input schema not used in a previous join pair.</p>
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 	<p>Optional: Click <b>Propose Join</b> to have Data Services generate a join expression.</p> <p>The SQL clause is automatically updated after you change the left source, right source, or join type.</p>
Join condition	<p>A join condition is required for each join pair.</p> <p>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.</p>

**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.

**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	<code>TABLE1.EMPNO &gt; 1000</code>
Multiple conditions using logical operators	<code>TABLE1.EMPNO &gt; 1000 OR TABLE2.EMPNO &lt; 9000</code>
Join conditions for inner joins only  <b>Note:</b> 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.	<code>TABLE1.EMPNO = TABLE2.EMPNO</code>

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.

#### 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.

#### 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.

#### 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 Topics**

- [Smart editor](#)

- [Functions and Procedures](#)
- [pushdown\\_sql](#)

#### 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 Topics**

- [To search in an input or output schema](#)

### 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.










#### 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.

 Mapping	SELECT	 FROM	WHERE	GROUP BY	ORDER BY	Advanced	Find
Input schema(s)	From	Join rank	Cache				
 ODS_MATERIAL	<input checked="" type="checkbox"/>	0	Yes				
 ODS_SALESITEM	<input checked="" type="checkbox"/>	0	Automatic				
 ODS_SALESORDER	<input checked="" type="checkbox"/>	0	Automatic				
 ODS_CUSTOMER	<input checked="" type="checkbox"/>	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.

#### 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	<p>The left source of a join.</p> <p>For the first join pair, select the input schema from the drop-down list of available schemas.</p> <p>For subsequent join pairs, the result of the previous join pair is taken as the left source and the schema cannot be specified.</p>
Join type	<p>The type of join.</p> <p>Valid values are Inner join and Left outer join.</p> <p>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.</p>
Right	<p>The right source of a join.</p> <p>Any input schema not used in a previous join pair.</p>
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 	<p>Optional: Click <b>Propose Join</b> to have Data Services generate a join expression.</p> <p>The SQL clause is automatically updated after you change the left source, right source, or join type.</p>
Join condition	<p>A join condition is required for each join pair.</p> <p>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.</p>

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 Topics**

- [Restricting left outer joins](#)
- [Designer Guide: Nested data, Operations on nested data, Overview of nested data and the Query transform](#)
- [Performance Optimization Guide: Other tuning techniques, Join ordering, Join rank settings](#)
- [Performance Optimization Guide: Using caches, Caching joins](#)

### 5.5.4.10.3 Join Examples

#### *Joins using two sources*

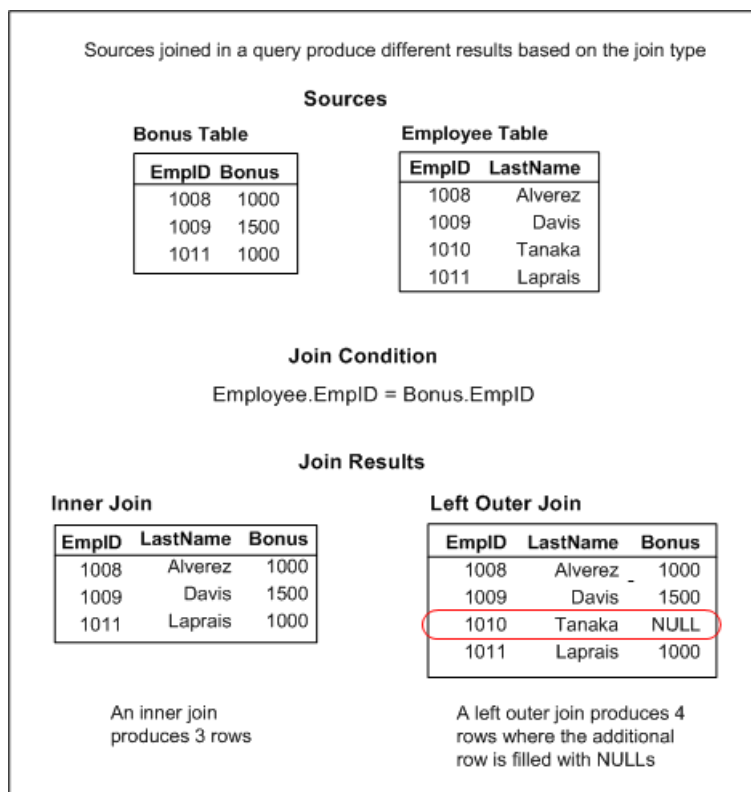
##### **About inner and left outer joins**

Sources joined in a Query transform produce different results based on the join type.

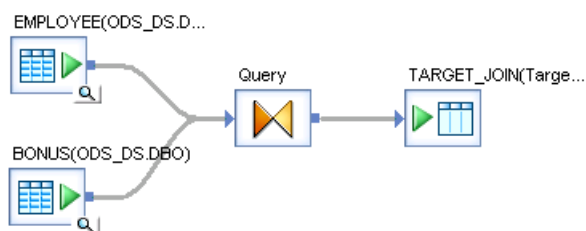
When joining two sources, an inner join returns rows from both sources where a match is found.

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:



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		Employee Table
EmpID	Bonus	EmpID LastName
1008	1000	1008 Alvarez
1009	1500	1009 Davis
1011	1000	1010 Tanaka
		1011 Laprais

Join Results		
Inner Join		
EmpID	LastName	Bonus
1008	Alvarez	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		Employee Table
EmpID	Bonus	EmpID LastName
1008	1000	1008 Alvarez
1009	1500	1009 Davis
1011	1000	1010 Tanaka
		1011 Laprais

Join Results		
Left Outer Join		
EmpID	LastName	Bonus
1008	Alvarez	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

### 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	Alvarez	01	1008	1000
02	Finance	1009	Davis	02	1009	1500
03	Sales	1010	Tanaka	01	1011	1000
04	Marketing	1011	Laprais	01		

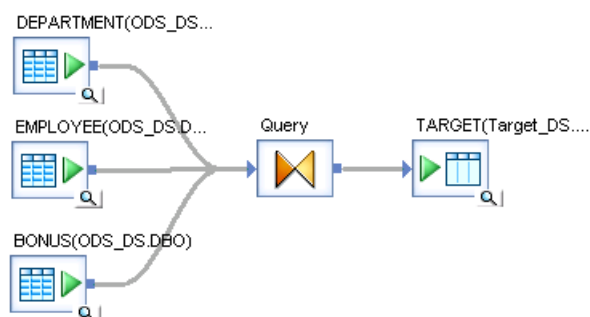
Join Results:

DeptID	Department	LastName	Bonus
1	Accounting	Alvarez	1000
1	Accounting	Tanaka	NULL
1	Accounting	Laprais	1000
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

### 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

#### 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		Sales Order Table
CUST_ID	Name	CUST_ID    Order Number
DT03	Pottery Ceramics	SA01    PT22221000
KT02	Major Resellers	SA01    PT22221001
SA01	Trusty Manufacturers	SA01    PT22221002
SA02	New Times	SA02    PT22221005
SA03	Popular Press	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'

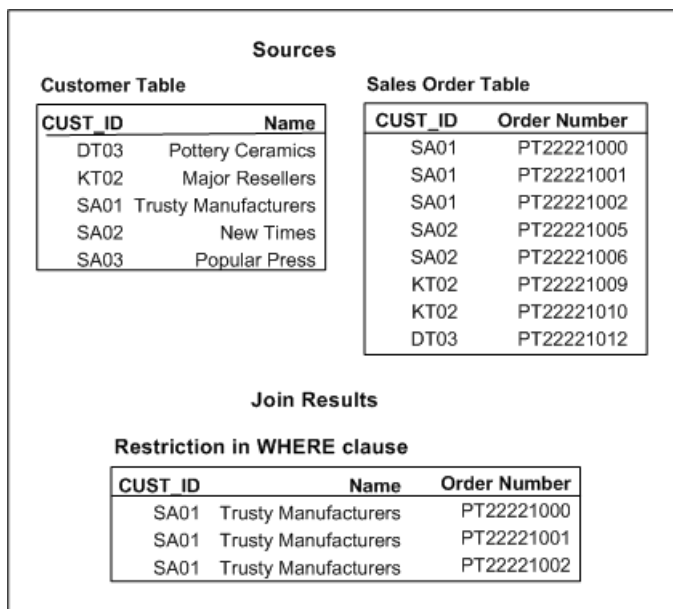
The SQL query is:

```
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.



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:

Mapping
SELECT
FROM
WHERE
GROUP BY

Functions...
Domains...
...

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')
```



#### 5.5.4.10.4 Viewing Optimized SQL

Before running a job, you can view the SQL code that SAP BusinessObjects 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.

#### 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	

Lookup function	Left Outer join
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

## 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.

### 5.5.5.1 Data inputs

None.

### 5.5.5.2 Editor

The Row\_Generation transform editor includes the target schema, and transform options.

### 5.5.5.3 Options

Option	Description
<b>Cache</b>	<p>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.</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• <b>Yes:</b> The source is always cached unless it is the outer-most source in a join.</li> <li>• <b>No:</b> The source is never cached.</li> </ul> <p>The default is <b>Yes</b>.</p> <p>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.</p>
<b>Join rank</b>	<p>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.</p> <p>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.</p> <p>Must be a non-negative integer. Default value is 0.</p>
<b>Row count</b>	<p>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.</p>
<b>Row number starts at</b>	<p>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.</p>

#### 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.

## 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.

### 5.5.6.1 Data inputs

None. This transform does not allow an input data set.

### 5.5.6.2 Options

Option	Description
<b>Array fetch size</b>	<p>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.</p> <p>This option is available for source tables from DB2, Informix, ODBC, Oracle, and SQL Server datastores.</p> <p>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.</p>
<b>Cache</b>	<p>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.</p> <p>Options are:</p> <ul style="list-style-type: none"><li>• <b>Yes</b>: The source is cached.</li><li>• <b>No</b>: The source is not cached.</li></ul> <p>The default is <b>Yes</b>.</p> <p>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.</p>

Option	Description
Database type	<p>Database type and versions available in the selected datastore.</p> <p>The <b>Database type</b> 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 <b>Database type</b> list box, edit the datastore configuration information using the Datastore Editor.</p> <p>The following describes how Data Services determines SQL text values. For more information about the Datastore Editor and its dialogs, see <a href="#">Datastore editor</a>.</p> <ul style="list-style-type: none"> <li>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 <b>Use values from</b> box in the Create New Configuration dialog box (a sub-dialog of the Datastore Editor).</li> </ul> <p><b>Note:</b></p> <p>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.</p> <ul style="list-style-type: none"> <li>If you also select the <b>Restore values if they already exist</b> 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 <b>Use values from</b> option.</li> </ul> <p>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.</p> <p><b>Note:</b></p> <p>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 <b>Update Schema</b> button. To support portability, add variables afterwards.</p>
Datastore	<p>The name of the datastore that Data Services uses to access the tables referred to in <b>SQL text</b>.</p>

Option	Description
<b>Join rank</b>	<p>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.</p> <p>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.</p> <p>Must be a non-negative integer. Default value is 0.</p>
<b>SQL text</b>	<p>The text of the SQL query. This string is passed to the database server.</p> <p>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.</p>
<b>Update schema</b>	<p>Click this option to automatically calculate and populate the output schema for the SQL SELECT statement.</p>

### 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 Topics

- [varchar](#)

## 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 reusable 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.

### 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. You 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 <b>Send to Pass</b> or <b>Send to Both</b> , 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 Topics

- [Validation Transform Options tab](#)

## 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 Topics

- [Validation Rules tab options](#)

## 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
- Preferable for more complex rules
- Can be created in SAP BusinessObjects 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:

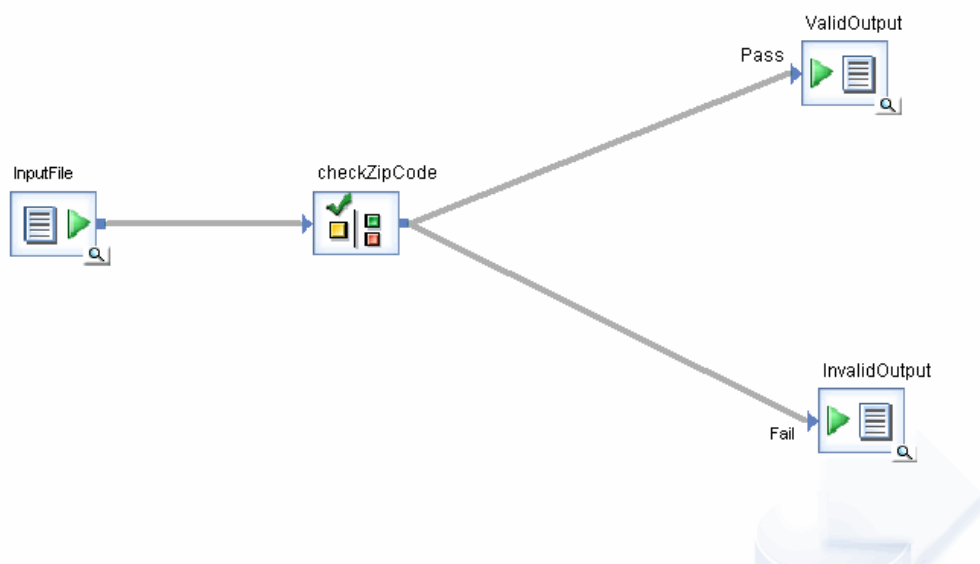
- Imported from Information Steward: These functions were created in Information Steward and cannot be edited in Data Services
- Locally created: Created and editable in Data Services

#### 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 (ellipses 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.

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:

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:

Ord.	OrderDate	Name	Address1	Address2	City	State	Zipcode	Phone	D
5003	2004.05.12	First Nat...	1 First St...	NULL	San Jose	CA	NULL	408-933-6821	B checkZipCode failed rule(s) Zipcode
5007	2004.05.16	First Nat...	1 First St...	NULL	San Jose	CA	951ab	408-933-6821	B checkZipCode failed rule(s) Zipcode
5007	2004.05.16	First Nat...	1 First St...	NULL	San Jose	CA	1234567	408-933-6821	B checkZipCode failed rule(s) Zipcode

### 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.

### *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 `ZipsValid` 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**.

#### **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.

### *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.

#### 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 description 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 <b>Validation Rules</b> 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 <b>Send to Pass</b> or <b>Send to Both</b> , 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 <b>Validation Rules</b> tab.

Option	Description
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	<p>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.</p> <p>Clear the <b>Score</b> check box to not include the binding in the Fail output. See the following example for more information about Score.</p>
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	<p>Select a condition (and usually in conjunction with an expression) to define the column-based rule. Available condition operators include:</p> <p>&lt;, &gt;, &lt;=, &gt;=, &lt;&gt;, =</p> <p><b>IS NULL, IS NOT NULL</b></p> <p><b>LIKE</b></p> <p><b>IN SET</b></p> <p><b>BETWEEN:</b> Use to specify a range of values.</p> <p><b>Match Pattern:</b> Lets you enter a pattern based on the Data Services match_pattern function.</p> <p><b>Exists in Table:</b> 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 <b>Exists in table</b> condition executes properly.</p> <p><b>Custom Condition:</b> 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.</p>

### 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 Topics

- [match\\_pattern](#)
- [lookup\\_ext](#)

## 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.

Option	Description						
Pass	When you choose a Pass type data output, the output schema is identical to the input schema.						
Fail	<p>When you choose a Fail type data output, Data Services adds the following columns to the Fail output schema: DI_ERRORACTION and DI_ERRORCOLUMNS.</p> <ul style="list-style-type: none"> <li>DI_ERRORACTION: This column indicates where Failed data was sent: <table border="1"> <thead> <tr> <th>Indicator</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </table> </li> </ul> <p><b>Note:</b> 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.</p> <ul style="list-style-type: none"> <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: <i>ValidationTransformName</i> failed rule(s): c1:c2</li> </ul> <p><b>Note:</b> 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.</p>	Indicator	Description	B	Sent to both Pass and Fail outputs.	F	Sent to the Fail output.
Indicator	Description						
B	Sent to both Pass and Fail outputs.						
F	Sent to the Fail output.						

Option	Description
RuleViolation	<p>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:</p> <ul style="list-style-type: none"> <li>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.</li> </ul> <p>To link the DI_ROWID to the actual data row, on the Validation Transform Options tab make sure the <b>Create column DI_ROWID on Validation_Fail</b> is selected, which includes DI_ROWID in the Fail output schema.</p> <ul style="list-style-type: none"> <li>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 <b>Score</b> selected in the Rule Editor will be added.</li> </ul>

### 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\_ERRORACTION and DI\_ERRORCOLUMNS) for the Fail output target at the top level of the schema only. Columns must be expressed with fully qualified names.

## 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.

### 5.5.8.1 Data inputs

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.

### 5.5.8.2 Data outputs

A single data set. The data set may be a hierarchical structure or row-based flat data.

### 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.

### 5.5.8.4 To search 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.

**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.

**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.

### 5.5.8.5 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
<b>Copy</b>	Columns, schemas	Stores a copy of the selected elements in the clipboard, leaving the elements in the input schema area.
<b>Find</b>	Anywhere in the input schema area	Locates an output element with the name or description you enter.
<b>Refresh</b>	Anywhere in the input schema area	Refreshes the display of the input schema area.
<b>Parent</b>	Columns	Selects the parent schema of the selected column.
<b>Collapse</b>	Columns, schemas	Collapses a selected schema or a selected column's parent schema (to facilitate viewing/navigation).
<b>Generate DTD</b>	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.
<b>Generate XML Schema</b>	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.
<b>Create File Format</b>	Schemas	Creates a file format from a relational table schema. All data types match those of the original table schema.
<b>Create HDFS File Format</b>	Schemas	Creates an HDFS file format from a relational table schema. All data types match those of the original table schema.
<b>Properties</b>	Columns, schemas	Displays the properties of the selected element.  <b>Note:</b> You cannot modify the element properties.

### 5.5.8.6 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.  <b>Note:</b> Data Services does not perform a complete validation during design, so the editor may not flag an incorrect mapping. For a complete validation, select <b>Validation &gt; Validate</b> .

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.

### 5.5.8.7 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	<p>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.</p> <p>If the cursor overlaps an existing column, you are prompted to insert above, insert below, remap column, or cancel.</p> <p><b>Note:</b> 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.</p>
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.

Command	Applicable elements	Effect
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.



### 5.5.8.8 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  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	<p>Specifies conditions that determine which instances of the target schema are output.</p> <p>Enter the conditions in SQL syntax, like a WHERE clause in a SQL SELECT statement. For example:</p> <pre>TABLE1.EMPNO = TABLE2.EMPNO AND TABLE1.EMPNO &gt; 1000 OR TABLE2.EMPNO &lt; 9000</pre> <p>Use the <b>Functions</b>, <b>Domains</b>, and <b>smart editor</b> buttons for help building expressions.</p>
DISTINCT	<p>Specifies the list of distinct columns from the input or output schema (if required).</p> <p><b>Restriction:</b> You can not mix input and output columns in the DISTINCT list.</p>
GROUP BY	Specifies how the output instances are combined (if required).

Tab	Description
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.

#### 5.5.8.8.1 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.

#### *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.
```

#### 5.5.8.8.2 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.

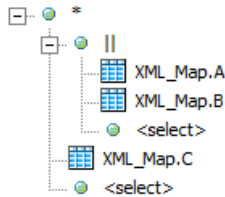
Data Services supports different kinds of joins in the iteration rule: INNER JOIN, LEFT OUTER JOIN, CROSS JOIN, and PARALLEL JOIN. PARALLEL JOIN is not a standard SQL join.

In the iteration rule tab, a hierarchical 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:

$(A \parallel B) * C$

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.

**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.

**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.

#### 5.5.8.8.3 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 function 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.

#### 5.5.8.8.4 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.

#### 5.5.8.8.5 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 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.

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.

#### 5.5.8.8.6 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.

#### 5.5.8.8.7 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
<b>Cache</b>	<p>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.</p> <p>Available values:</p> <ul style="list-style-type: none"> <li>• <b>Automatic</b> (default)</li> <li>• <b>Yes</b></li> <li>• <b>No</b></li> </ul> <p>Cache specified in the Advanced tab of the XML_Map transform editor overrides any cache specified in a source.</p>

#### 5.5.8.8 Find tab

Use the Find tab to search for a specific word or term in the input schema or the output schema.

#### Related Topics

- [To search in an input or output schema](#)

### 5.5.8.9 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
<pre>Employee  -departmentID  -departmentName  -employeeName  -employedDate</pre>	<pre>Company  -department (*)    -departmentID    -departmentName    -employee (*)      -employeeName      -employedDate  -totalEmployees</pre>

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)
```

### 5.5.8.10 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.

Source	Target
<pre>purchaseOrders  -purchaseOrder (*)    -sellerParty      -sellerID      -sellerName    -buyerParty      -buyerID      -buyerName    -orderLine      -item (*)        -name        -quantity        -unitPrice        -currency</pre>	<pre>item  -sellerID  -buyerID  -itemName  -totalSales  -currency</pre>

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 `purchaseOrders` 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.

### 5.5.8.11 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.

Source	Target
<pre>catalog  -book    -Name    -Price    -Quantity    -Author      -firstName      -lastName      -Author_nt_1        -street      -city      -state      -zip</pre>	<pre>authors  -Author    -Name      -firstName      -lastName    -Address      -Author_nt_1        -street      -city      -state      -zip    -book      -Name      -Price      -Quantity      -totalSales</pre>

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.

- 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.

- 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
```

- 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
<code>authors.Author.Name.firstName</code>	<code>catalog.book.Author.firstName</code>
<code>authors.Author.Name.lastName</code>	<code>catalog.book.Author.lastName</code>
<code>authors.Author.Address.Author_nt_1</code>	<code>catalog.book.Author.Author_nt_1</code>
<code>authors.Author.Address.Author_nt_1.street</code>	<code>catalog.book.Author.Author_nt_1.street</code>
<code>authors.Author.Address.city</code>	<code>catalog.book.Author.city</code>
<code>authors.Author.Address.state</code>	<code>catalog.book.Author.state</code>
<code>authors.Author.Address.zip</code>	<code>catalog.book.Author.zip</code>

- 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
```

- 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
```

- 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
```

## 5.6 Text Data Processing transforms

### 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.

#### 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.

##### 5.6.1.1.1 Common

The **Common** option group includes a setting to run the transform as a separate process.

Option	Description
<b>Run as Separate Process</b>	<p>Yes: Splits the transform into a separate process.</p> <p><b>Note:</b> The Entity Extraction transform is always run as a separate process. You cannot change the value of this option.</p>

#### 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
<b>Language</b>	<p>Specifies the language for processing your content. You must select a language from the list of available languages displayed alphabetically in the drop-down menu.</p> <p><b>Note:</b> 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.</p> <p>The default location for the language directory is: <code>&lt;LINK_DIR&gt;/TextAnalysis/languages</code>.</p>
<b>Filter By Entity Types</b>	<p>Specifies a list of entity types (supported by the selected language) to use for filtering the extraction output.</p> <p>By default, a drop-down menu showing '...' displays. Clicking this launches the "Ordered Options Window - [Filter By Entity Types - Option] " dialog.</p> <p>Select or remove one or more entity types from the list of available entity types for that language.</p> <p><b>Note:</b> Entity type support varies among languages.</p>

#### Related Topics

- [Text Data Processing Language Reference Guide: Language Modules Reference](#)

#### 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
<b>Dictionary Only</b>	<p>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.</p> <p><b>Note:</b> If you select this option, the extraction output will not include any predefined entities. Along with this option, if you also select the <b>Rule</b> option, the extraction output will include entities and facts defined in the rules along with entities from the specified dictionaries.</p>
<b>Advanced Parsing</b>	<p>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.</p> <p>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.</p>
<b>Processing Timeout</b>	<p>Use this option to stop processing the content after a set amount of time. By default, the <b>Processing Timeout</b> option is set to 60 seconds. The Processing Timeout value can be one of the following:</p> <ul style="list-style-type: none"> <li>-1 indicates no timeout should be enforced.</li> <li>&gt;=1 indicates the amount of time (in seconds) after which processing should abort.</li> </ul>



Option	Description
<b>Document Properties</b>	<p>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.</p> <p>Document properties are name-value pairs. The Entity Extraction transform extracts only the following document properties for binary documents:</p> <ul style="list-style-type: none"> <li>• APP_NAME: The name of the software that was used to create the document</li> <li>• APP_VERSION: The version of the software that was used to create the document</li> <li>• AUTHOR: The name of the person who created the document</li> <li>• COMPANY: The name of the company that owns the document</li> <li>• TITLE: The title of the document</li> <li>• DATE_CREATED: The date on which the document was created</li> </ul> <p>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:</p> <ul style="list-style-type: none"> <li>• ID: The entity ID of the document property</li> <li>• SOURCE: DOC_PROPERTY</li> <li>• TYPE: The name of the document property</li> <li>• STANDARD_FORM or SOURCE_FORM: The value of the document property</li> <li>• CONVERTED_TEXT: The textual content of the binary document</li> </ul> <p>Any other output columns are not applicable to DOC_PROPERTY extraction rows, and have their value set to -1.</p>

### Related Topics

- [Text Data Processing Language Reference Guide: Language Modules Reference](#)

#### 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
<b>Dictionary</b>	<p>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.</p> <p>Once the entry is duplicated, change the duplicate dictionary file by selecting the dictionary to use from the directory structure.</p>
<b>Dictionary file</b>	<p>Use the <b>Browse</b> option under the drop-down menu to select a valid, compiled dictionary file to use for extraction.</p> <p><b>Note:</b> 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).</p>
<b>Filter By Entity Types</b>	<p>Specifies a list of entity types (defined in the selected dictionary) to use for filtering the extraction output.</p> <p>By default, a drop-down menu showing '...' displays. Clicking this launches the "Ordered Options Window - [Filter By Entity Types - Option] " dialog.</p> <p>Select or remove one or more entity types from the available entity list.</p>

### Related Topics

- [Text Data Processing Extraction Customization Guide: Using Dictionaries](#)
- [Designer Guide: Dictionary overview](#)
- [Designer Guide: Executing Jobs, Changing Job Server options, To use mapped drive names in a path](#)

#### 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
<b>Rule</b>	<p>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.</p> <p>Once the entry is duplicated, change the duplicate rule file by selecting the rule you want to use from the directory structure.</p>

Option	Description
<b>Rule File</b>	<p>Use the <b>Browse</b> option under the drop-down menu to select a valid, compiled rule file to use for extraction.</p> <p>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).</p> <p><b>Note:</b> A rule file typically contains multiple rules. You can use the rule filtering option to select a specific rule in a rule file.</p>
<b>Filter By Rule Names</b>	<p>Specifies a list of rule names (defined in the selected rule file) to use for filtering the extraction output.</p> <p>By default, a drop-down menu showing '...' displays. Clicking this launches the "Ordered Options Window - [Filter By Rule Names - Option] " dialog.</p> <p>Select or remove one or more rules from the filtering list.</p>

### Related Topics

- [Text Data Processing Extraction Customization Guide: Using Rules](#)
- [Designer Guide: Rule Overview](#)
- [Designer Guide: Executing Jobs, Changing Job Server options, To use mapped drive names in a path](#)

## 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).

Name	Data Type	Description
TEXT_ID	Long, int, or varchar	<p>Optional field. Unique identifier to be used for tracing the content in case of an error.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>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 <code>TEXT</code> input field is be used as the content identifier.</li> <li>When a <code>varchar</code> or <code>long</code> column is mapped to the <code>TEXT_ID</code> input field and a value used to construct an error message contains more than 1K bytes, the value will be truncated to 1K.</li> </ul>

### 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	<p>Represents a parent-child relationship between entities and/or facts. This value is unique within the scope of the processed input text.</p> <p><b>Note:</b></p> <p>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.</p>
PARENT_ID	int	<p>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.</p>
STANDARD_FORM	varchar (2000)	<p>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 <code>TYPE</code> column.</p> <p><b>Note:</b></p> <p>The standard form and the source form for an entity are often the same.</p>

Generated field name	Data Type	Description
TYPE	varchar (255)	<p>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 <code>PERSON</code> entity and “car” as a <code>COMMON_VEHICLE/LAND</code> entity subtype.</p> <p><b>Note:</b> “/” is used as a separator to identify subtypes.</p>
SOURCE_FORM	varchar (2000)	The name of an entity, fact, or subfact as mentioned in the input text.
SOURCE	varchar (10)	<p>The origin of an entity or fact. Meaning, how the match was determined, based on one of the following:</p> <ul style="list-style-type: none"> <li>• <code>SYSTEM</code> - indicating that the entity was matched using the system files.</li> <li>• <code>DICTIONARY</code> - indicating that the entity was matched using a dictionary.</li> <li>• <code>RULE</code> - indicating that the entity or fact was matched using an extraction rule file.</li> </ul>
OFFSET	int	The character offset of an entity or a fact in the <code>CONVERTED_TEXT</code> field.
LENGTH	int	The character length of an entity or a fact in the <code>CONVERTED_TEXT</code> field.
PARAGRAPH_ID	int	A unique identifier of the paragraph in the <code>CONVERTED_TEXT</code> field containing the entity or fact.
SENTENCE_ID	int	A unique identifier of the sentence in the <code>CONVERTED_TEXT</code> field containing the entity or fact.
CONVERTED_TEXT	long	<p>The content text representation in UTF-16 encoding of the input text.</p> <p><b>Note:</b> When the <code>CONVERTED_TEXT</code> 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.</p>

**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	StrongNegativeSentiment	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

ID	PAR ENT_ID	STAN DARD_FORM	TYPE	SOURCE_FORM	SOURCE
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	WeakPosi tiveSentiment	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_VE HICLE/LAND	car	SYSTEM
17	-1	from \$2500 to \$4500	CURRENCY	from \$2500 to \$4500	SYSTEM
18	-1	Mr. Jones	PERSON	Mr. Jones	SYSTEM
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	GeneralRe quest	would like	RULE
22	19	Green's compre- hensive cover- age	Topic	Green's comprehensive coverage	RULE
23	-1	Green	PROP_MISC	Green	SYSTEM
24	-1	other competi- tors	COM- MON_PERSON /GROUP	other competitors	SYSTEM

ID	PARENT_ID	STANDARD_FORM	TYPE	SOURCE_FORM	SOURCE
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 Topics**

- [Designer Guide: XML extraction and parsing for columns](#)



# 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.

## 6.1 About functions

### 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 BusinessObjects Data Services does not support functions that include tables as input or output parameters, except functions imported from SAP Applications.

### 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
<b>Aggregate</b>	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 <b>Group By</b> tab of a query.  Can be called only from within a Query transform—not from custom functions or scripts.
<b>Iterative</b>	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.
<b>Stateless</b>	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.

### 6.1.3 Arithmetic in date functions

Data Services performs some implicit data type conversions on date, time, datetime, and interval values.

#### Related Topics

- [Date arithmetic](#)

### 6.1.4 Including functions in expressions

You can use functions in the following:

- Transforms (Query , Case, SQL)
- 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 Topics

- [Smart editor](#)

## 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.

### 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 Topics**

- [Operation of a function](#)
- [Including functions in expressions](#)

## 6.2 Built-in functions

SAP BusinessObjects 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 Topics**

- [Performance Optimization Guide: Degree of parallelism and functions](#)
- [Designer Guide: Distributed data flow execution](#)
- [Catch error functions](#)

## 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 Topics**

- [About procedures](#)
- [Designer Guide: Imported stored function and procedure information](#)

## 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.

Function	Category	Description
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.
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.
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.

Function	Category	Description
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 Dataflow 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.
exec	System	Sends a command to the operating system for execution.

Function	Category	Description
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 integer 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.



Function	Category	Description
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.
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.

Function	Category	Description
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).
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.

Function	Category	Description
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.
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.

Function	Category	Description
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_process_chain_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.
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.

Function	Category	Description
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.
sysptime	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.
wait_for_file	Miscellaneous	Returns the existing files that match the input file pattern.

Function	Category	Description
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 Topics

- [Catch error functions](#)

## 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

<i>num</i>	The source number.
------------	--------------------

**Example:**

Function	Results
<code>abs(12.12345)</code>	12.12345
<code>abs(-12.12345)</code>	12.12345

---

### 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**

<code>original_date</code>	Specify the starting year.month.date.
<code>months_to_add</code>	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
<code>add_month('1990.12.17', 1)</code>	'1991.01.17'
<code>add_month('2001.10.31', 4)</code>	'2002.2.28'

### 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**

<i>input_string</i>	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
<code>ascii('AaC')</code>	65

---



### 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

<i>value_list</i>	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.

### 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

<i>base64-encoded input</i>	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 parameter is not required when the input data type is blob.

#### Related Topics

- [base64\\_encode](#)

## 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

<i>input_data</i>	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 Topics

- [base64\\_decode](#)

### 6.3.7 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

<i>expression</i>	Input expression that needs to be cast to target data type.
<i>data_type</i>	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

Cast type compatibility matrix										
From / To	Date	Date time	Decimal	Double	Int	Interval	Real	Time	Time stamp	Varchar
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	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 6-15: Example

Input	Output
<code>cast('20.3', 'decimal(3,1)')</code>	20.3

### 6.3.8 chr

Converts a decimal ASCII code to a character.

#### Syntax

```
chr (integer_expression)
```

#### Return Value

ASCII character

#### Where

<code>integer_expression</code>	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 BusinessObjects 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
<code>chr (65)</code>	'A'

---

**6.3.9 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**

<i>num</i>	The source number.
------------	--------------------

**Example:**

Function	Results
<code>ceil(12.12345)</code>	13.00000
<code>ceil(12)</code>	12
<code>ceil(-12.223)</code>	-12.000

---

### 6.3.10 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")
```

---

### 6.3.11 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**

<i>column</i>	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)
```

**6.3.12 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



Name	Region	Country
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 Topics**

- [Designer Guide: Distributed data flow execution](#)

### 6.3.13 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**

<i>ds_name</i>	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
```

---

### 6.3.14 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
```

---

### 6.3.15 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()]')
```

---

## 6.3.16 datastore\_field\_value

Retrieves the value of a specified datastore field.

**Syntax**

```
datastore_field_value(ds_name, field_name)
```

**Return Value**

varchar

**Where**

<i>ds_name</i>	The name you enter when you create the datastore.
<i>field_name</i>	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
<code>datastore_field_value('mssql', 'sql_server_database')</code>	'DBUser'

---

## 6.3.17 date\_diff

Returns the difference between two dates or times.

**Syntax**

```
date_diff(date1, date2, fmt_str)
```

**Return Value**

int

**Where**

<i>date1, date2</i>		
<i>fmt_str</i>	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
<code>date_diff(start_date, sysdate(), 'D')</code>	The number of days between the date in the column <code>start_date</code> and today's date.
<code>date_diff(start_time, systime(), 'M')</code>	The number of minutes between the time in the column <code>start_time</code> and the current time.

---

## 6.3.18 date\_part

Extracts a component of a given date.

**Syntax**

```
date_part(in_date, fmt_str)
```

**Return Value**

int

**Where**

<i>in_date</i>	The input date.	
<i>fmt_str</i>	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
<code>date_part('1990.12.31', 'YY')</code>	1990
<code>date_part('1991.01.17 23:44:30', 'SS')</code>	30

---

### 6.3.19 `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**

<i>date1</i>	The source date.
--------------	------------------

This function extracts the day component from the date value.

**Example:**

Function	Results
<code>day_in_month(to_date('Jan 22, 1997','mon dd, yyyy'))</code>	22
<code>day_in_month(to_date('02/29/1996','mm/dd/yyyy'))</code>	29
<code>day_in_month(to_date('1996.12.31','yyyy.mm.dd'))</code>	31

---

### 6.3.20 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**

<i>date1</i>	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
<code>day_in_week(to_date('Jan 22, 1997','mon dd, yyyy'))</code>	3 (Wednesday)
<code>day_in_week(to_date('02/29/1996','mm/dd/yyyy'))</code>	4 (Thursday)
<code>day_in_week(to_date('1996.12.31','yyyy.mm.dd'))</code>	2 (Tuesday)

---

### 6.3.21 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**

<i>date1</i>	The source date.
--------------	------------------



**Example:**

Function	Results
<code>day_in_year(to_date('Jan 22, 1997', 'mon dd, yyyy'))</code>	22
<code>day_in_year(to_date('02/29/1996', 'mm/dd/yyyy'))</code>	60
<code>day_in_year(to_date('1996.12.31', 'yyyy.mm.dd'))</code>	366 (1996 was a leap year.)

**6.3.22 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, Sybase (for 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

Datastore Types	Possible db_type() Return Value
SAP Applications	SAP
SAP BW Source	SAP
SAP BW Target	SAP_BW
SAP Master Data Services	HANA
Siebel	DB2, Microsoft_SQL_Server, or Oracle

**Where**

<i>ds_name</i>	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 BusinessObjects 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}');
```

**6.3.23 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
Sybase ASE	Currently supported versions
Sybase IQ	Currently supported versions
Teradata	Currently supported versions
'''	An empty string is returned for any other database type

**Where**

<i>ds_name</i>	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}');
```

### 6.3.24 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 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 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 Sybase ASE as the Database Type.

#### Syntax

```
db_database_name(ds_name)
```

#### Return Value

varchar

#### Where

<i>ds_name</i>	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 }');
```

### 6.3.25 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

<code>ds_name</code>	The datastore name that you entered when you created the datastore.
<code>alias_name</code>	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');
```

### 6.3.26 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**

<i>condition_and_expression_list</i>	<p>A comma-separated list of one or more pairs that specify a variable number of conditions. Each pair contains one <i>condition</i> and one <i>expression</i> separated by a comma. You must specify at least one <i>condition</i> and <i>expression</i> pair.</p> <p>The <i>condition</i> evaluates to TRUE or FALSE.</p> <p>The <i>expression</i> is the value that the function returns if the <i>condition</i> evaluates to TRUE.</p>
<i>default_expression</i>	<p>An expression that the function returns if none of the conditions in <i>condition_and_expression_list</i> evaluate to TRUE. You must specify a <i>default_expression</i>.</p>

**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
<pre>decode((COUNTRY = 'FRANCE'), 'French', (COUNTRY = 'GERMANY'), 'German', (COUNTRY = 'ITALY'), 'Italian', (COUNTRY = 'TAIWAN'), 'China', (COUNTRY = 'USA'), 'America', (COUNTRY IS NULL), 'Unknown', 'Others')</pre>	<p>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.</p>

### 6.3.27 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

<i>encryptedinput_string</i>	A varchar input string to be decrypted.
<i>passphrase</i>	A varchar character string.
<i>key_length_in_bits</i>	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 workflow.

### 6.3.28 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**

<i>Encrypted_input_string</i>	A varchar input string to be decrypted.
<i>Passphrase</i>	A varchar character string with at least one character.
<i>Salt</i>	A varchar that must be exactly eight ASCII characters.
<i>Key_length_in_bits</i>	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 workflow.

### 6.3.29 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**

<i>input_str</i>	The source string to encode.
<i>alternate</i>	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 <i>return_if_empty</i> 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 <i>return_if_empty</i> is set. When the parameter is null or invalid (non-numeric), it is defaulted to 0.

<code>return_if_empty</code>	<p>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.</p> <p>When input is empty, there is no primary or secondary encodings. When <code>return_if_empty=0</code>, then return null. When <code>return_if_empty=1</code>, then return the empty string.</p>
------------------------------	---

### 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
<code>Print(double_metaphone('Hello',0,0);</code>	Prints the double metaphone of the word "Hello."
<code>double_metaphone(\$VAR, 1,1);</code>	If the string stored in \$VAR does not have encoding available, then return the original string.
<code>double_metaphone(\$VAR,'a','b');</code>	Returns the primary double metaphone encoding or the variable \$VAR when the primary encoding does not exist.

## 6.3.30 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**

<i>input_string</i>	A varchar input string to be encrypted.
<i>passphrase</i>	A varchar character string.
<i>key_length_in_bits</i>	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 workflow.

**6.3.31 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

<i>Input_string</i>	A varchar input string to be encrypted.
<i>Passphrase</i>	A varchar character string.
<i>Salt</i>	A varchar that must be exactly eight ASCII characters.
<i>Key_length_in_bits</i>	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 workflow.

## 6.3.32 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 Topics**

- [Catch error functions](#)

**6.3.33 error\_message**

Returns the error message of the caught exception.

**Syntax**

```
error_message()
```

**Return value**

varchar 512

**Related Topics**

- [Catch error functions](#)

**6.3.34 error\_number**

Returns the error number of the caught exception.

**Syntax**

```
error_number()
```

**Return value**

int

**Related Topics**

- [Catch error functions](#)

**6.3.35 error\_timestamp**

Returns the timestamp of the caught exception.

**Syntax**

```
error_timestamp()
```

**Return value**

timestamp

**Related Topics**

- [Catch error functions](#)

**6.3.36 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**

<i>command_file</i>	<p>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.</p> <p>The <i>command_file</i> can be a Windows batch file, a UNIX shell script, or a binary executable. To run other interpreted scripts, the <i>command_file</i> must be the name of the command interpreter (e.g., 'perl') and the script must be the first parameter in the <i>parameter_list</i>.</p>
<i>parameter_list</i>	<p>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 ('').</p>
<i>flag</i>	<p>An integer that specifies what information appears in the return value string, and how to respond upon error—how to respond if <i>command_file</i> cannot be executed or exits with a nonzero operating system return code.</p>

Table 6-52: 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: <ul style="list-style-type: none"> <li>• The command never produces output</li> <li>• The calling job does not need output</li> </ul>
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 "

Flag	If successful, returns:	On error:	Notes:
256	NULL string	NULL string	<p>Use this flag if you want your program to run independently of Data Services.</p> <p>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.</p> <p>Raises an exception (System function failure) if the program cannot be launched (e.g., program file not found).</p>

### 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', 'RemoteMachineNameCommandToRunRemotelyCmdArg1CmdArg2', 0);
```

```
exec('rsh', 'RemoteBox -lRemoteUserRemoteCommandCmdArg', 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', 'RemoteMachineNameremcmdWrapper.batCmdArg1CmdArg2', 8);
```
  - ```
exec('rsh', 'RemoteBox -lRemoteUser /usr/acta/remcmdWrapperCmdArg', 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, ARG_TBL.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);
```

---

### 6.3.37 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
<code>xml_column_name</code>	The name of the input column that contains the XML text. The column data type must be varchar.
<code>schema_dtd_name</code>	<p>The name of the DTD or XML Schema format that describes the incoming XML text.</p> <p>You must import the metadata for this format into Data Services. Data Services displays the format names in the <b>Formats</b> tab of the object library. The input value must be a constant since Data Services needs to know the output schema at design time.</p>
<code>enable_validation</code>	<p>Enable a comparison of the incoming XML data against the format you specify for <code>schema_dtd_name</code>. 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 <code>1</code> to validate. Enter <code>0</code> to disable the validation option.</p> <p>Validation is useful during development. It provides a more precise error if a problem occurs with the incoming XML.</p>

### 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 Topics

- [Designer Guide: Nested Data, Scenario 1](#)

### 6.3.38 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

<i>file_path</i>	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')
```

---

### 6.3.39 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**

<i>start_year_date</i>	The first month and day of a fiscal year. Use this format: 'mm.dd'.
<i>in_date</i>	The date you want to convert. Use any valid datetime.

**Example:**

Function	Results
<code>fiscal_day('03.01', '1999.04.20')</code>	51

---

## 6.3.40 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**

<i>num</i>	The source number.
------------	--------------------

**Example:**

Function	Results
<code>floor(12.12345)</code>	12.00000
<code>floor(12)</code>	12
<code>floor(-12.223)</code>	-13.000

**6.3.41 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**

<i>expression_list</i>	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
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

**6.3.42 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()
```

---

### 6.3.43 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.

### 6.3.44 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**

<i>domain_name</i>	Fully qualified domain name, including the database owner if required. For example:  <i>datastorename.owner.domain</i>  <i>datastorename..domain</i>
<i>domain_value</i>	The constant value for which you want to return a description.

**Example:**

Function	Results
<code>get_domain_description('psoft..ACTION', 'ADL')</code>	"Additional"

---

### 6.3.45 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**

<i>variable_name</i>	The name of the environment variable. The name must be surrounded by single quotes.
----------------------	---

**Example:**

Function	Results
<code>getenv('TMP')</code>	C:\Temp

---

### Related Topics

- [is\\_set\\_env](#)

## 6.3.46 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
```

## 6.3.47 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**

<i>filename</i>	The file name and path relative to the current Job Server. Enter a file name as a relative path or an absolute path.
<i>attribute</i>	One of the following attributes, which must be in single quotes: <code>date_created</code> , <code>date_modified</code> , or <code>size</code> .

**Example:**

Function	Results
<code>get_file_attribute('/order', 'date_created')</code>	'2004:09:15:20:25:00'  The format in this example is YYYY:MM:DD:HH24:MM:SS.
<code>get_file_attribute(c:\database\order, 'size')</code>	'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.

**6.3.48 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_fcfl1e004c968.txt
```

**Note:**

A monitor log is referred to as a trace log in the Designer.

---

### 6.3.49 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_fcfl004c968.txt
```

### 6.3.50 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

<i>expression_list</i>	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'
3	NULL

---

### 6.3.51 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()]');
```

---

### 6.3.52 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

<i>condition</i>	An expression that evaluates to TRUE or FALSE.
<i>true_branch</i>	An expression that the function returns if <i>condition</i> evaluates to TRUE.
<i>false_branch</i>	An expression that the function returns if <i>condition</i> 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
<pre>ifthenelse (ZIP &lt; 94000, 'SOUTH', 'NORTH')</pre>	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.

### 6.3.53 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

<i>input_string</i>	The source string.
<i>index_string</i>	The character sequence sought in <i>input_string</i> .
<i>start</i>	The position where the function starts searching in <i>input_string</i> for the character sequence contained in <i>index_string</i> .  <i>Start</i> should be a positive number between 1 and the length of <i>input_string</i> .

#### 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
<code>index('Accounting Department', 'DEPARTMENT', 1)</code>	0
<code>index('Accounting Department', 'Department', 1)</code>	12

### 6.3.54 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**

<i>value</i>	The string to be modified.
<i>locale</i>	Optional parameter that converts the string to the specified locale. <b>Note:</b> ISO 639 language code and ISO 3166 country code formats are supported.

**Example:**

Function	Results
<code>init_cap('Data Services')</code>	'Data Services'
<code>init_cap(StreetAddress)</code>	Writes the value, for example '1234 west washington school road', in column <code>StreetAddress</code> as '1234 West Washington School Road'.
<code>Print(Init_cap('have a nice day -hyphen +plus _underscore \slash \$dollar *star @at tab mIXedWORD UPPER lower !punctuations 1234digits'));</code>	Have A Nice Day -Hypen +Plus _Under-score \Slash \$Dollar *Star @At Tab Mixedword Upper Lower !Punctuations 1234digits
<code>init_cap(LastName, 'tr')</code>	<p>The value in column <code>LastName</code> 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.</p> <p>It is also converted to the Turkish locale, using the ISO 639 language code.</p>

**Limitations**

- The function is pushed down to Oracle databases only.
- You cannot use this function in an ABAP data flow.

**Related Topics**

- [ISO 639 language list](#)
- [ISO 3166 Country Code list](#)

**6.3.55 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

<i>input_interval</i>	The value of type <code>interval</code> to convert.	
<i>interval_type</i>	A string describing the format of the interval. Choose from the following values:	
	D	Days
	H	Hours
	M	Minutes
	S	Seconds

#### Example:

Function	Results
<code>interval_to_char(start_date - sysdate(), 'd')</code>	The number of days between the date in the column <code>start_date</code> and today's date.
<code>interval_to_char(start_time - systime(), 'm')</code>	The number of minutes between the time in the column <code>start_time</code> and the current time.

## 6.3.56 is\_group\_changed

Returns 1 if the group is changed, 0 otherwise.

#### Syntax

```
is_group_changed(expression)
```

#### Return Value

Integer

**Where**

<i>expression</i>	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
<code>is_group_changed(state,city)</code>	<code>1,0,1,0,0,1,1</code>

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

**6.3.57 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**

<i>variable_name</i>	The name of the environment variable. The name must be surrounded by single quotes.
----------------------	---

**Example:**

Function	Results
<code>is_set_env('MODE')</code>	Returns 1 if the MODE variable has already been set; returns 0 if the MODE variable has not been set.

**6.3.58 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**

<i>input_expression</i>	<p>The expression to be validated.</p> <p>If the expression does not resolve to a value of data type <code>varchar</code>, you will see a warning that the value has been converted to a <code>varchar</code>.</p>
-------------------------	--

<i>date_format</i>	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
<code>is_valid_date (Orders.SubmitDate, 'mm/dd/yyyy')</code>	Tests whether the string <code>Orders.SubmitDate</code> can be converted to a calendar date with the <code>mm/dd/yyyy</code> date format.

**Related Topics**

- [date](#)

**6.3.59 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**

<i>input_expression</i>	The expression to be validated.	
<i>datetime_format</i>	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
<code>is_valid_datetime (Orders.Received,'mm/dd/yyyy hh24:mi:ss')</code>	Tests whether the string <code>Orders.Received</code> can be converted to the <code>mm/dd/yyyy hh24:mi:ss</code> date-time format.

**Related Topics**

- [datetime](#)

### 6.3.60 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

<i>input_expression</i>	The expression to be validated.
<i>decimal_format</i>	<p>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: '#,###,###.##'.</p> <p>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:</p> <pre>is_valid_decimal (Stocks.Price_difference, '-###.##')</pre>

#### Example:

Function	Results
<pre>is_valid_decimal (Orders.Price, '#,###,###.##')</pre>	Tests whether the string <code>Orders.Price</code> can be converted to decimal format.



### 6.3.61 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

<i>input_expression</i>	The expression to be validated.
<i>double_format</i>	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
<pre>is_valid_double (Product.Weight, '###.###')</pre>	Tests whether the string <code>Product.Weight</code> can be converted to double format.

### 6.3.62 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

<i>input_expression</i>	The expression to be validated.
<i>int_format</i>	The format specifying the thousands separator of the input expression. For example, to specify an integer format, use the following string: <b>###.###</b> . 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
<code>is_valid_int (QuarterResults.Volume, '###.###')</code>	Tests whether the string <code>QuarterResults.Volume</code> can be converted to the <code>###.###</code> integer format.

## 6.3.63 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**

<i>input_expression</i>	The expression to be validated.
<i>real_format</i>	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
<code>is_valid_real (QuarterResults.Mean, '#,###.####')</code>	Tests whether the string <code>QuarterResults.Mean</code> can be converted to real format.

**6.3.64 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**

<i>input_expression</i>	The expression to be validated.	
<i>time_format</i>	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	2-digit hour of the day (00-23)
	MI	2-digit minute (00-59)
	SS	2-digit second (00-59)

**Example:**

Function	Results
<code>is_valid_time (Orders.ReceivedTime, 'hh24:mi:ss')</code>	Tests whether the string <code>Orders.ReceivedTime</code> can be converted to the <code>hh24:mi:ss</code> datetime format.

**6.3.65 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**

<i>table_name</i>	<p>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.</p> <p>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.</p>
-------------------	---

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)
```

**6.3.66 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**

<i>date1</i>	The value of type <code>date</code> or <code>datetime</code> to test.
--------------	---

**Example:**

Function	Results
<code>isweekend(hire_date)</code>	Tests whether the date in <code>hire_date</code> is a Saturday or Sunday.
<code>isweekend(SYSDATE)</code>	Tests whether today is a Saturday or Sunday.

---

### 6.3.67 `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()]');
```

---

### 6.3.68 `Job_Run_ID`

Retrieves the job run ID for the current job execution.

**Syntax**

```
job_run_id()
```

**Return value**

Varchar (20)

**6.3.69 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**

<i>date1</i>	The source value of type <code>date</code> or <code>datetime</code> .
--------------	---

**Example**

Function	Results
<pre>julian(to_date('Apr 19, 1997', 'mon dd, yyyy'))</pre>	729436

**6.3.70 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**

<i>input_julian</i>	An integer representing the Julian value to be converted.
---------------------	---

**Example:**

Function	Results
<code>julian_to_date(Julian_Date)</code>	Converts the number indicated by <code>Julian_Date</code> to its date value.

---

### 6.3.71 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.

**Syntax**

```
key_generation (table, key_column, key_increment)
```

**Return value**

int

The column value found to meet the function requirements.

**Where**

<i>table</i>	The full path name of the file or full database specification of the table in which the <i>key_column</i> is located. Enclose this value in single quotation marks.
<i>key_column</i>	A column with existing keys from which this function determines the largest existing key value. Enclose this value in single quotation marks.



<code>key_increment</code>	The integer increment used between key values this function generates.
----------------------------	--

**Example:**

Function	Results
<code>key_generation('target_ds.dbo.sales','order_number',1)</code>	Looks for the last key value in the <code>order_number</code> column in the sales database and returns the largest value plus one.

---

### 6.3.72 last\_date

Returns the last date of the month for a given date.

**Syntax**

```
last_date(in_date)
```

**Return Value**

date

**Where**

<code>in_date</code>	The date for which the last date of the month is to be calculated.
----------------------	--

**Example:**

Function	Returns
<code>last_date('1990.10.01')</code>	<code>'1990.10.31'</code>

### 6.3.73 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**

<i>expression_list</i>	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 an 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

---

### 6.3.74 length

Returns the number of characters in a given string.

**Syntax**

```
length (value)
```

**Return value**

integer

The number of characters in *value*.

**Where**

<i>value</i>	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

**6.3.75 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_expression_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

<i>input</i>	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');
```

### 6.3.76 ln

Returns the natural logarithm of the given numeric expression.

**Syntax**

```
ln(numeric_expression)
```

**Return Value**

Float

**Where**

<i>numeric_expression</i>	Any numeric expression.
---------------------------	-------------------------

**Details**

Returns the natural logarithm of the given numeric expression. Return value is NULL if input is negative.

**Example:**

Function	Results
<code>ln(5.436563656918)</code>	1.693147

---

### 6.3.77 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_validation, 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	<p>The name of the DTD or XML Schema that you want Data Services to use to format the XML text in the output column.</p> <p>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 <code>is_top_level_element</code> for an example of specifying an only child.</p> <p><b>Note:</b> You can generate an XML Schema or DTD from an NRDM schema using Data Services.</p>
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 <code>schema_dtd_name</code> .
xml_header	<p>If the value specified is NULL, the header of the output XML has UTF-8 in the header. The default header generated is:</p> <pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt;</pre> <p>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.</p>
replace_null_string	If this value is not null, Data Services replaces NULL values with the specified string.

<code>is_top_level_element</code>	<p>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.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>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 <code>nested_table_name</code>, it matches the root element of the schema, so set the value of <code>is_top_level_element</code> to 1.</li> <li>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 as the <code>nested_table_name</code>, because it does not match the root element of the schema allEmployees, set the value for <code>is_top_level_element</code> 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.</li> </ul>
<code>max_size</code>	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 Topics**

- [Designer Guide: Nested Data, XML extraction and parsing for columns](#)
- [Designer Guide: Nested Data, Generating DTDs and XML Schemas from an NRDM schema](#)

**6.3.78 log**

Returns the base-10 logarithm of the given numeric expression.

**Syntax**

```
log (num)
```



**Return Value**

Float

**Where**

<i>num</i>	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
<code>log(100.000)</code>	2.000000

---

## 6.3.79 long\_to\_vchar

Converts a data type value of a given column from long to varchar.

**Syntax**

```
long_to_vchar(column_name, max_size, start_position)
```

**Return value**

varchar

**Where**

<i>column_name</i>	The name of the column containing the long data type.
<i>max_size</i>	The maximum size for the converted data in the table column.

<i>start_position</i> (Optional)	Starting character position from which data is converted. The default start position is 1. A negative number 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 Topics**

- [Designer Guide: XML extraction and parsing for columns, Scenario 1](#)

## 6.3.80 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**

<i>lookup_table</i>	<p>The table or file that contains the result or value you are looking up (<i>result_column</i>). The <i>compare_column</i> is also located in this table. Use a fully qualified table name, including the datastore, owner, and table name. For example:</p> <pre>oracle_ds.TIGER.sales</pre> <p>You might need to put the owner in quotes, particularly if you use lower case letters.</p>
---------------------	--

<i>result_column</i>	The column containing the values you want to retrieve. This column is in the <i>lookup_table</i> .
<i>default_value</i>	The value returned when there is no matching row in the <i>lookup_table</i> .
<i>cache_spec</i>	<p>The caching method the lookup operation uses. List within single quotes. There are three possible settings:</p> <p>NO_CACHE: Reads values from the <i>lookup_table</i> for every row without caching values.</p> <p>PRE_LOAD_CACHE: Loads the <i>result_column</i> and <i>compare_column</i> into memory after applying filters but before executing the function.</p> <p>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.</p> <p>DEMAND_LOAD_CACHE: Loads <i>result_column</i> and <i>compare_column</i> values into memory as the function identifies them.</p> <p>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.</p> <p>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.</p>
<i>compare_column</i>	The column in the <i>lookup_table</i> that the function uses to find a matching row. When the function reads a varchar column in the <i>lookup_table</i> , it does not trim trailing blanks.
<i>expression</i>	<p>The value that the function searches for in the <i>compare_column</i>. This can be a simple column reference, such as a column found in both a source and the <i>lookup_table</i>. This can also be a complex expression given in terms of constants and input column references.</p> <p>When <i>expression</i> refers to a unique source column, you do not need to include a table name qualifier. If <i>expression</i> is from another table or is not unique among the source columns, you need a table name qualifier.</p> <p>If <i>expression</i> is an empty string, the function searches for a zero-length varchar value in the <i>compare_column</i>.</p> <p>The function ignores trailing blanks in comparisons of <i>expression</i> and values in <i>compare_column</i>.</p>

**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.

**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."

### 6.3.81 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

<i>lookup_table</i>	<p>The table, file, or memory datastore that contains the result(s) or value(s) you are looking up (<i>result_column_list</i>). If the <i>lookup_table</i> is a database table, use the <i>datastore.owner.table</i> format. For example:</p> <pre>ERP_ds.OWNER.EMPLOYEES</pre> <p>If the <i>lookup_table</i> is a flat file, use the <i>file_ds.filename</i> format. For example:</p> <pre>delim."c:/temp/employees"</pre> <p>To substitute a variable for a file name, replace the data inside the double quotes, for example <code>delim."\$employees"</code>. 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 BusinessObjects Data Services can pass in a different file name each time it calls <code>lookup_ext</code>. For example, you can call <code>lookup_ext</code> 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.</p> <p>If the <i>lookup_table</i> is a memory datastore table, use the <i>memory_ds.table</i> format. For example:</p> <pre>mem_ds..employees</pre>
---------------------	---

<i>cache_spec</i>	<p>The caching method the <code>lookup_ext</code> operation uses. You can select one of the following cache specifications:</p> <ul style="list-style-type: none"> <li>• <b>NO_CACHE</b>—Reads values from the <i>lookup_table</i> for every row without caching values.</li> <li>• <b>PRE_LOAD_CACHE</b>—Loads the <i>return_column_list</i>, <i>compare_column</i> (see <i>condition_list</i>), and <i>orderby_column_list</i> into memory after applying constant filters and before executing the function.</li> </ul> <p>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.</p> <ul style="list-style-type: none"> <li>• <b>DEMAND_LOAD_CACHE</b>—Loads <i>return_column_list</i>, <i>compare_column</i> (see <i>condition_list</i>), and <i>orderby_column_list</i> into memory as the function identifies them.</li> </ul> <p>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.</p> <p>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.</p>
<i>return_policy</i>	<p>Use <i>return_policy</i> 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 <code>lookup_ext</code> function to treat NULL as the smallest value instead of the largest value.</p>
<i>return_column_list</i>	<p>A comma-separated list containing the names of output columns in the <i>lookup_table</i>.</p> <p>For a given output column in the lookup table, select the <b>Expression?</b> check box in the <code>lookup_ext</code> 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.</p>
<i>default_value_list</i>	<p>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.</p> <p>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.</p> <p>If <i>default_value_list</i> 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.</p>

<code>condition_list</code>	<p>A list of triplets that specify lookup conditions. Each set in a triplet contains <code>compare_column</code>, <code>compare_operator</code> (&lt;, &lt;=, &gt;, &gt;=, =, !=, IS, IS NOT, ~), and <code>compare_expression</code>.</p> <p>The <code>compare_column</code> is from the <code>lookup_table</code>. It is compared against <code>compare_expression</code> to compute the output row.</p> <p>The <code>compare_expression</code> is written in terms of constants, variables, and columns in the calling data flow or scripts. While it cannot contain column reference from the <code>lookup_table</code>, it can be a simple constant, variable, or column reference or a complex expression involving arithmetic operations and function calls.</p> <p>Use compare operators IS and IS NOT to examine <code>compare_column</code> against the NULL constant. When you use IS or IS NOT as the compare operator, <code>compare_expression</code> must contain the NULL constant. When you use other compare operators against a <code>compare_expression</code> 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:</p> <ul style="list-style-type: none"> <li>• <code>mp (pattern)</code>—Indicates the match_pattern type of pattern syntax</li> <li>• <code>mr (pattern)</code>—Indicates the match_regex type of pattern syntax</li> <li>• <code>ms (pattern)</code>—Indicates the match_simple type of pattern syntax</li> </ul> <p>If <code>compare_expression</code> is an empty string, the function looks up a zero-length varchar value in the lookup table. The function ignores trailing blanks in the <code>compare_expression</code>.</p> <p>If you create more than one set of triplets, all triplets are implicitly joined with the AND operator to compute the final lookup condition.</p> <p>Example:</p> <pre>[c1,     '=', 10, c2, '&lt;', query.a, c3, '&gt;=', lower (query.name) ]</pre>
<code>orderby_column_list</code>	<p>A comma-separated list of column names from the <code>lookup_table</code>. Working together with <code>return_policy</code>, the <code>orderby_column_list</code> 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 <code>orderby_column_list</code> and choosing a row to return using the MIN/MAX <code>return_policy</code>.</p> <p>The <code>orderby_column_list</code> is optional. If you leave it blank, the orderby columns match the output columns.</p> <p>Examples:</p> <p>[c1,c3,c4] — Sorts the rows using column values in c1, c3, c4.</p> <p>[] — Indicated an empty list, which is a placeholder for specifying subsequent parameters.</p>



<code>output_variable_list</code>	<p>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.</p> <p>This parameter is optional unless more than one output column appears in the <code>return_column_list</code>. In the case of more than one output column, output variables must be equal in number to output columns.</p> <p>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.</p> <p>Example: [\$a,\$b,\$c]</p>
<code>sql_override</code>	<p>This parameter, available as a button called <b>Custom SQL</b> 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 <code>return_column_list</code>, <code>condition_list</code>, and <code>orderby_column_list</code>.</p> <p>Any valid SQL SELECT statement is permitted and may contain references to other tables besides the <code>lookup_table</code> to specify inner and outer joins. This parameter can only be specified when the <code>lookup_table</code> is a database table.</p> <p>If you specify this parameter with the cache specification of NO_CACHE, the software executes the <code>sql_override</code> query each time the function is called.</p> <p>If you specify this parameter with the cache specification of PRE_LOAD_CACHE, only the first <code>sql_override</code> query is executed to populate the lookup cache. All subsequent SQL statements are ignored after the lookup cache is built.</p> <p>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.</p> <p>EXAMPLE:</p> <pre>['select out1,   out2,compare1,compare2,orderby1,orderby2 from lookuptbl,othertbl where c1=10 and lookuptbl.c2=othertbl.c2']</pre> <p>Also, when you use <code>sql_override</code> in NO_CACHE mode, <code>lookup_ext</code> will dynamically execute the SQL if you pass in a dynamic SQL statement in the form of a variable.</p>

The SET options include:

run_as_separate_process	<p>Select (set to <code>yes</code>) 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 <code>no</code>.</p> <pre>SET ("run_as_separate_process"='yes')</pre>
output_cols_info	<p>Identifies whether an output column contains an expression (when the <b>Expression</b> check box is selected). The default is <code>no</code>. The syntax is as follows:</p> <pre>SET ("output_cols_info"='&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;output_cols_info&gt; &lt;col index="1" expression="yes"/&gt; &lt;/output_cols_info&gt;')</pre> <p>The first column begins with index value of 1.</p>

### 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, 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 Topics

- [Designer Guide: Data Flows, Lookup tables and the lookup\\_ext function](#)
- [match\\_pattern](#)
- [match\\_regex](#)
- [match\\_simple](#)
- [Performance Optimization Guide: Run as a separate process option](#)

## 6.3.82 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

<i>lookup_table</i>	<p>The table or file that contains the result or value you are looking up (<i>result_column</i>). The <i>sequence_column</i> and <i>compare_column</i> are also located in this table. Use a fully qualified table name, including the datastore, owner, and table name. For example:</p> <p>ERP_ds.OWNER.EMPLOYEES</p> <p>The <i>lookup_table</i> is cached automatically for the operation of the function.</p>
---------------------	---

<i>result_column</i>	<p>The column containing the values you want to retrieve. This column is in the <i>lookup_table</i>.</p> <p>When the column contains varchar values, the function does not trim trailing blanks.</p>
<i>default_value</i>	The value returned when there is no matching row in the <i>lookup_table</i> .
<i>sequence_column</i>	<p>The column in <i>lookup_table</i> 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, <i>sequence_column</i> is the EFFDT column, which indicates when the data in the row became effective.</p>
<i>sequence_expression</i>	<p>The value the function searches for in the <i>sequence_column</i> 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 <i>sequence_expression</i>.</p> <p>If <i>sequence_expression</i> is an empty string, the function looks up a zero-length varchar value in the lookup table.</p>
<i>compare_column</i>	The column in the <i>lookup_table</i> that the function uses to find a matching row.
<i>expression</i>	<p>The value that the function searches for in the <i>compare_column</i>. This can be a simple column reference, such as a column found in both a source and the <i>lookup_table</i>. This can also be a complex expression given in terms of constants and input column references.</p> <p>When <i>expression</i> refers to a unique source column, you do not need to include a table name qualifier. If <i>expression</i> is from another table or is not unique among the source columns, you need a table name qualifier.</p> <p>If <i>expression</i> is an empty string, the function looks up a zero-length varchar value in the lookup table.</p> <p>The function ignores trailing blanks in comparisons of <i>expression</i> and values in <i>compare_column</i>.</p>

**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, '12.31.1999', 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.

---

### 6.3.83 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**

<i>value</i>	The string to be modified.
<i>locale</i>	Optional parameter that converts the string to the specified locale. <b>Note:</b> ISO 639 language code and ISO 3166 country code formats are supported.

**Example:**

Function	Results
<code>lower('Accounting101')</code>	'accounting101'
<code>upper((LastName,1,1))   lower(substr(LastName,2,LENGTH(LastName)))</code>	The value in column <code>LastName</code> with the first letter uppercase and the rest of the value lowercase. Note that this example does not account for two-word last names.
<code>lower(LastName, 'tr')</code>	The value in column <code>LastName</code> is converted to all lowercase. It is also converted to the Turkish locale, using the ISO 639 language code.

**Related Topics**

- [ISO 639 language list](#)
- [ISO 3166 Country Code list](#)

**6.3.84 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**

<i>input_string</i>	The string source.
<i>size</i>	An integer value indicating the number of characters in the return string.
<i>pad_string</i>	A character or set of characters that this function concatenate to <i>input_string</i> .

**Example:**

Function	Results
<code>lpad('Tanaka', 15, '')</code>	'            Tanaka '
<code>lpad(last_name, 25, '')</code>	The value in the column <code>last_name</code> , padded with spaces to 25 characters on the left or truncated to 25 characters.

---

### 6.3.85 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

<i>input_string</i>	The string source.
<i>size</i>	An integer value indicating the number of characters in the return string.
<i>pad_string</i>	A logical character or set of logical characters that this function concatenates to the <i>input_string</i> .

#### Example:

Function	Results
<code>lpad('Tanaka', 15, ' ')</code>	' Tanaka'
<code>lpad(last_name, 25, ' ')</code>	The value in the column <code>last_name</code> , padded with spaces to 25 characters on the left or truncated to 25 characters.

The extended function yield results different from the basic function when "char1" or "char2" contains multibyte characters. For instance, let "𐀀" be a Chinese ideograph, a double-byte character which occupies two cells on a display device or printer, then:

Function	Input	Output
<code>lpad</code>	<code>('abc𐀀', 10, '𐀀')</code>	" 𐀀𐀀abc"
<code>lpad</code>	<code>('abcd', 10, '𐀀')</code>	"𐀀𐀀𐀀𐀀abcd"
<code>lpad</code>	<code>('ab𐀀', 4, '𐀀')</code>	"𐀀ab"
<code>lpad_ext</code>	<code>('abc𐀀', 10, '𐀀')</code>	" 𐀀𐀀𐀀𐀀𐀀𐀀abc"
<code>lpad_ext</code>	<code>('abcd', 10, '𐀀')</code>	"𐀀𐀀𐀀𐀀𐀀𐀀abcd"
<code>lpad_ext</code>	<code>('ab𐀀', 4, '𐀀')</code>	"𐀀𐀀ab"

### 6.3.86 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**

<i>input_string</i>	The string to be modified.
<i>trim_string</i>	The characters to remove from <i>input_string</i> .

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
<code>ltrim('Marilyn', ' ')</code>	'Marilyn'
<code>ltrim('ABCABCD', 'ABC')</code>	'D'
<code>ltrim('ABCABCD', 'EFG')</code>	'ABCABCD'
<code>ltrim('ABCDEABCDE', 'ABC')</code>	'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.

---

### 6.3.87 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**

<i>input_string</i>	The string to be modified.
---------------------	----------------------------

**Example:**

Function	Results
<pre>ltrim_blanks('  Marilyn')</pre>	<pre>'Marilyn'</pre>
<pre>ltrim_blanks(last_name)</pre>	The value contained in the column <i>last_name</i> , with all leading blanks and control characters removed.

---

## 6.3.88 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**

<i>input_string</i>	The string to be modified.
---------------------	----------------------------

**Example:**

Function	Results
<code>ltrim_blanks_ext(' Marilyn')</code>	'Marilyn'
<code>ltrim_blanks_ext(last_name)</code>	The value contained in the column last_name, with all leading blanks removed.

**6.3.89 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**

<i>recipients_list</i>	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.
<i>subject</i>	A string containing the subject of the e-mail. This string can be empty.
<i>message</i>	A string containing the e-mail message. This string can be empty.
<i>number_of_trace_lines</i>	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.

<code>number_of_error_lines</code>	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 BusinessObjects 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
<pre>\$myvar = mail_to('admin@company.com', 'Out of memory error in the SalesFact job. Please fix the error before running recovery job.', ' ', 10, 10)</pre>	The message is sent to one recipient.
<pre>\$myvar = mail_to('admin@company.com, manager@compa ny.com', 'Out of memory error:'    systime(), 'Out of memory error while running the data flow:'    \$dataflow_name    ' in the job:'    \$job_name    ',', 10, 20)</pre>	<p>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 includes a blank on the beginning of the next string. You can also concatenate a string with a single blank.</p> <p>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 <b>Functions</b> button to construct a mail_to statement.</p>

**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 Topics**

- [Microsoft Help and Support](#)

**6.3.90 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: <ul style="list-style-type: none"> <li>• Ll—Lowercase letter (for example Latin, Greek, Cyrillic, Armenian, Deseret, and archaic Georgian).</li> <li>• Lt—Title letters (for example Latin capital letter D with small letter Z).</li> <li>• Lm— Modifier letter (for example acute accent, grave accent).</li> <li>• Lo—Other letters (including Chinese, Japanese, and so on).</li> </ul>
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

Example Value	Pattern string
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"	<pre>if (match_pattern('15014', '![12]9999') &lt;&gt; 0)   print('matched'); else   print('not matched');</pre>	Function prints "not matched".
To match a zip code except one that begins with 1 or 2.	"[!12]9999"	<pre>if (match_pattern('55014', '![12]9999') &lt;&gt; 0)   print('matched'); else   print('not matched');</pre>	Function prints "matched".
To process only customer phone numbers that fit the same pattern.	"999-999-9999"	<pre>WHERE MATCH_PATTERN(CUS- TOMER.PHONE_NUM, '999-999- 9999') &lt;&gt; 0</pre>	Phone numbers that do not match the pattern throw error 0.

### Related Topics

- [literal](#)

## 6.3.91 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**

<code>input_string</code>	String to be matched. Supports UNICODE characters.
<code>regular_expression_pattern</code>	<p>Pattern you want to find in a whole input string. Substring matches are not supported.</p> <p>Provide the pattern in regular expression format with a varchar data type.</p>
<code>flags</code>	<p>Allows you to specify additional behavior that you want to occur while Data Services searches the <code>input_string</code> for pattern matches.</p> <p>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.</p>

You can use the following regular expression patterns in the pattern parameter:

Character	Description
<code>\a</code>	Match a BELL, \u0007.
<code>\A</code>	Match at the beginning of the input. Differs from <code>^</code> in that <code>\A</code> will not match after a new line within the input.
<code>\b</code> , outside of a [Set]	Match if the current position is a word boundary. Boundaries occur at the transitions between word ( <code>\w</code> ) and non-word ( <code>\W</code> ) characters, with combining marks ignored. For better word boundaries, see ICU Boundary Analysis.
<code>\b</code> , within a [Set]	Match a BACKSPACE, \u0008.
<code>\B</code>	Match if the current position is not a word boundary.
<code>\cX</code>	Match a control-X character.
<code>\d</code>	Match any character with the Unicode General Category of Nd (Number, Decimal Digit).
<code>\D</code>	Match any character that is not a decimal digit.

Character	Description
\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.
\s	Match a white space character. White space is defined as [\t\n\r\f\rp{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{Li}\p{Lu}\p{Lt}\p{Lo}\p{Nd}].

Character	Description
\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.

Operator	Description
*	Match 0 or more times. Match as many times as possible.
+	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.

Operator	Description
<code>{n,+}</code>	Match at least n times. Possessive match.
<code>{n,m}+</code>	Match between n and m times. Possessive match.
<code>( ... )</code>	Capturing parentheses. Range of input that matched the parenthesized subexpression is available after the match.
<code>(?: ... )</code>	Non-capturing parentheses. Groups the included pattern, but does not provide capturing of matching text. Somewhat more efficient than capturing parentheses.
<code>(?&gt; ... )</code>	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 "(?>"
<code>(?# ... )</code>	Free-format comment ( <code>?# comment</code> ).
<code>(?= ... )</code>	Look-ahead assertion. True if the parenthesized pattern matches at the current input position, but does not advance the input position.
<code>(?! ... )</code>	Negative look-ahead assertion. True if the parenthesized pattern does not match at the current input position. Does not advance the input position.
<code>(?&lt;= ... )</code>	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).
<code>(?&lt;! ... )</code>	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).
<code>(?ismx-ismx: ... )</code>	Flag settings. Evaluate the parenthesized expression with the specified flags enabled or disabled.
<code>(?ismx-ismx)</code>	Flag settings. Change the flag settings. Changes apply to the portion of the pattern following the setting. For example, <code>(?i)</code> 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	<p>If set, a "." in a pattern will match a line terminator in the input text. By default, it will not.</p> <p>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.</p>

#### 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})"	<pre>match_regex (pho_number, '([0-9]{3}-[0-9]{3}-[0-9]{4})', NULL)</pre>
To match a string that starts with "topicA" regardless of case.	"topicA.*"	<pre>match_regex (subject, 'topicA.*', CASE_INSENSITIVE)</pre>
To check a string against a complex pattern and print result to trace log.	"XXX)\$@&*xxX9999xX9#"	<pre>if (match_pattern('JJD') \$@&amp;*hhN8922hJ7#', 'XXX)\$@&amp;*xxX9999xX9#') &lt;&gt; 0)   print ('matched'); else   print ('not matched');</pre> <p>The result for this call is "matched".</p>

#### Related Topics

- [literal](#)

### 6.3.92 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

<i>input_string</i>	String to be matched. Supports UNICODE characters.
<i>pattern_string</i>	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:  <pre>{ABC+;XYZ*}</pre> If the data matches either ABC+ or XYZ*, the result is TRUE.
<>	NOT operator. Specify the pattern after the <>. Example:  <pre>&lt;&gt;pattern</pre>



{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]	<pre>if (match_simple('ACCT14', 'ACCT[1..5000]')) &lt;&gt; 0 print ('matched'); else print('not matched');</pre>	Function prints "matched".

## 6.3.93 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**

<i>value_list</i>	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.

## 6.3.94 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**

<i>value_list</i>	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.
- 

## 6.3.95 mod

Returns the remainder when one number is divided by another.

### Syntax

```
mod(number1, number2)
```

### Return Value

Depends on the input numbers.

### Where

<i>number1</i>	Number to be divided.
<i>number2</i>	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
<code>mod(100.10,10);</code>	0.100000

---

### 6.3.96 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**

<i>date1</i>	The source date.
--------------	------------------

**Example:**

Function	Results
<code>month(to_date('Jan 22, 1997', 'mon dd, yyyy'))</code>	1
<code>month(to_date('3/97', 'mm/yy'))</code>	3

---

### 6.3.97 num\_to\_interval

Converts a numeric value to an interval.

### Syntax

```
num_to_interval(number1, format)
```

### Return value

interval

The converted interval.

### Where

<i>number1</i>	The value of type <code>int</code> , <code>real</code> , <code>decimal</code> , or <code>numeric</code> to convert.	
<i>format</i>	A string describing the format of the interval. Choose from the following values:	
	D	Days
	H	Hours
	M	Minutes
	S	Seconds

### Example:

Function	Results
<code>num_to_interval(elapsed_days, 'D')</code>	The value from the column <code>elapsed_days</code> converted to an interval of days.
<code>start_time + num_to_interval(elapsed_seconds, 'S')</code>	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').

### 6.3.98 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

<i>expression1</i>	The value to be tested for NULL.
<i>replacement_value</i>	The value to replace <i>expression1</i> if <i>expression1</i> is NULL. <i>replacement_value</i> must be the same data type as <i>expression1</i> .

#### Example:

Function	Results
<pre>nvl(modification_date, sysdate())</pre>	If the column <code>modification_date</code> for a row hasn't been set, this function inserts today's date.
<pre>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'))</pre>	Both expressions are determined by the result of lookup functions.

---

### 6.3.99 power

Returns the value of the given expression to the specified power.

**Syntax**

```
power (num, num)
```

**Return Value**

Float data type

**Where**

<i>num</i>	Numeric expression representing base number.
<i>num</i>	Numeric expression representing power.

**Details**

Returns the value of the given expression to the specified power.

**Example**

Function	Results
<code>power (2.2, 3) ;</code>	10.648000

**6.3.100 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**

<i>expression</i>	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

---

**6.3.101 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**

<i>input_string</i>	The message to be written to the trace log.
---------------------	---

**Example:**

Function	Results
<pre>print('Reached decision point for running full or incremental data flows')</pre>	Writes "Reached decision point for running full or incremental flows" to trace log and returns <i>input_string</i> .
<pre>print('The date is: [\$start_date]')</pre>	Writes "The date is 2000.06.03" to trace log and returns <i>input_string</i> .
<pre>print('[\$month_sal*12]')</pre>	Writes "48000" to trace log and returns <i>input_string</i> .
<pre>print('Total Sal is: [\$month_sal*12]');</pre>	Writes "Total Sal is: 48000" to trace log and returns <i>input_string</i> .
<pre>print('The return value from the SQL() function is &gt; [\$y]');</pre>	Writes "The return value from the SQL() function is > 23456" to trace log and returns <i>input_string</i> .

**6.3.102 pushdown\_sql**

Allows you to create dynamic WHERE clauses.

## Syntax

```
pushdown_sql (datastore, input_string)
```

## Return Value

None.

## Where

<i>datastore</i>	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.
<i>input_string</i>	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
<code>pushdown_sql ('EC_DS', '{XML_IN.QUERY_REQUEST}')</code>	<p>Data Services includes the following WHERE clause in the SQL SELECT statement:</p> <pre>WHERE TYPE = 'MOUNTAIN' and PRICE &lt; 1500</pre>

**6.3.103 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**

<i>date1</i>	The source date.
--------------	------------------

**Example:**

Function	Results
<code>quarter(to_date('Jan 22, 1997', 'mon dd, yyyy'))</code>	1
<code>quarter(to_date('5/97', 'mm/yy'))</code>	2

---

### 6.3.104 **raise\_exception**

Calling this function causes an exception to be generated.

#### **Syntax**

```
raise_exception(error_msg)
```

#### **Return Value**

int

Returns '1' always.

#### **Where**

<code>error_msg</code>	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.'))
```

---

### 6.3.105 **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**

<i>error_msg</i>	The string which will be written to the Job Server's error log.
<i>exit_code</i>	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))
```

---

## 6.3.106 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
<code>100 * rand()</code>	A random number between 0 and 100.

**6.3.107 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,  $x_n = (ax_{n-1} + b) \bmod m$  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 specifying 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**

<i>seed</i>	(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
<code>100 * rand_ext()</code>	A random number between 0 and 100.

---

### 6.3.108 **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**

<i>in_str</i>	The input string that you are changing. If NULL, return will be NULL.
<i>search_str</i>	String to search for. If <i>search_string</i> is NULL, varchar is returned and will be the same as <i>in_str</i> .
<i>replace_str</i>	If <i>replace_string</i> is omitted or NULL, all occurrences of <i>search_string</i> are removed.

**Example:**

Function	Result
<code>replace_substr('a penny saved is a penny earned', 'penny', 'million')</code>	<code>'a million saved is a million earned'</code>

---

### 6.3.109 **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_occurrences)
```

**Return Value**

varchar

**Where**

<code>in_str</code>	The input string that you are changing. If NULL, returns NULL.
---------------------	--



search_str	<p>String to search for:</p> <p>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.</p> <p>You can also represent special characters using the escape character '/'. The following are supported.</p> <p>/a Bell (alert)</p> <p>/b Backspace</p> <p>/f Formfeed</p> <p>/n New line</p> <p>/r Carriage return</p> <p>/t Horizontal tab</p> <p>/v Vertical tab</p> <p>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'.</p> <p>If search_str is NULL, Data Services returns a varchar with the data in in_str.</p>
replace_str	<p>String that replaces search_str. If replace_string is omitted or NULL, all occurrences of search_str are removed.</p>
start_at_occurrence	<p>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.</p>
number_of_occurrences	<p>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.</p>

**Example:**

Function	Result
Replace 'a' with 'B' starting from second occurrence and replaces two occurrences: <pre>replace_substr_ext('ayyyayyyayyyayyy', 'a', 'B', 2, 2)</pre>	'ayyyByyyByyyayyy'
Search a string containing 'a' followed by a new line and replace it with 'B' starting from second occurrence and replaces two occurrences: <pre>replace_substr_ext('ayyya&lt;newline&gt;yyya&lt;newline&gt;yyyayyy', 'a/n', 'B', 2, 2)</pre>	'ayyyByyyByyyayyy'
Search a string containing 'a' followed by a new line and replace it with 'B' starting from second occurrence and replaces two occurrences: <pre>replace_substr_ext('ayyya&lt;newline&gt;yyya&lt;newline&gt;yyyayyy', 'a/x000a', 'B', 2, 2)</pre>	'ayyyByyyByyyayyy'

**6.3.110 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()]')
```

**6.3.111 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**

<i>num1</i>	The source number.
<i>precision</i>	An integer indicating the number of decimals in the result. If <i>precision</i> is negative, digits left of the decimal point are rounded.

**Example:**

Function	Results
<code>round(120.12345, 2)</code>	120.12
<code>round(120.12999, 2)</code>	120.13
<code>round(120, -2)</code>	100
<code>round(120.123, 5)</code>	120.12300

---

### 6.3.112 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

## Where

<i>input_string</i>	The source string.
<i>size</i>	An integer value indicating the number of characters in the resulting string.
<i>pad_string</i>	A character or set of characters that this function concatenates to <i>input_string</i> .

Function	Results
<code>rpad('Tanaka',15,' ')</code>	'Tanaka '
<code>rpad(last_name,25,' ')</code>	The value in the column <code>last_name</code> , padded with spaces to 25 characters, or truncated to 25 characters.

### 6.3.113 rpad\_ext

**Note:**

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

The new string.

## 2012-11-22

<i>input_string</i>	The source string.
<i>size</i>	An integer value indicating the number of characters in the resulting string.
<i>pad_string</i>	A character or set of characters that this function concatenates to <i>input_string</i> .

**Example:**

Function	Results
<code>rpadd_ext('Tanaka',15,' ')</code>	'Tanaka'
<code>rpadd_ext(last_name,25,' ')</code>	The value in the column <code>last_name</code> , padded with spaces to 25 characters, or truncated to 25 characters.

The extended function yield results different from the basic function when "char1" or "char2" contains multibyte characters. For instance, let "𐀀" be a Chinese ideograph, a double-byte character which occupies two cells on a display device or printer, then:

Function	Input	Output
rpadd	("abc", 10)	"abc "
rpadd	("abcd", 10, "x")	"abcdxxxxxx"
rpadd	("ab", 4)	"ab"
rpadd_ext	("abc", 10)	"abc "
rpadd_ext	("abcd", 10, "x")	"abcdxxxxxxxx"
rpadd_ext	("ab", 4)	"ab"

### 6.3.114 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

<i>input_string</i>	The string to be modified.
<i>trim_string</i>	The characters to remove from <i>input_string</i> .

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
<code>rtrim('Marilyn ', ' ')</code>	'Marilyn'
<code>rtrim('ZABCABC', 'ABC')</code>	'Z'
<code>rtrim('ZABCABC', 'EFG')</code>	'ZABCABC'

**6.3.115 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**

<i>input_string</i>	The string to be modified.
---------------------	----------------------------

If the length of the modified string becomes zero after trimming, the function returns "" (empty string).

**Example:**

Function	Results
<code>rtrim_blanks('Marilyn ')</code>	'Marilyn'
<code>rtrim_blanks(last_name)</code>	The value contained in the column <code>last_name</code> with trailing blanks removed.

### 6.3.116 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

<i>input_string</i>	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 <code>last_name</code> with trailing blanks and control characters removed.

---

### 6.3.117 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 BusinessObjects 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**

<i>datastore</i>	Specifies the datastore name. You can specify either a constant string or a substitution parameter.  The data type is varchar(64).
<i>open_hub_table</i>	Specifies the Open Hub Destination table. You can specify either a constant string or a substitution parameter.  The data type is varchar(30).
<i>process_chain</i>	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).
<i>\$logid</i>	(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).
<i>\$ReturnTxt</i>	(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.

### 6.3.117.1 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.

### 6.3.118 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

<i>datastore</i>	Specifies the datastore name. You can specify either a constant string or a substitution parameter.  The data type is varchar(64).
<i>destination</i>	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).
<i>status</i>	Specifies the read status. Possible values are either a variable or one of the following string constants: <ul style="list-style-type: none"> <li>'X' for Read Successful</li> <li>Any other value indicates that the Read failed.</li> </ul> The data type is varchar(1).
<i>\$returntxt</i>	(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);
```

### 6.3.119 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**

<i>sr_table_spec</i>	<p>A constant string that specifies the search and replace table or file. It has three possible valid forms:</p> <p><i>Datastore.owner.table</i> — Specifies a database table containing the search and replace values.</p> <p><i>Fileformat.filename</i> — Specifies a fixed or delimited file containing the search and replace values.</p> <p>NULL — Used when the search and replace is performed using custom SQL or an internal table defined in the SET options.</p>
<i>search_column</i>	<p>Specifies the column name in the table or file containing the search values. If <i>sr_table_spec</i> is an internal table, this should be set to NULL.</p> <p>The data contained in this column must be of the varchar type.</p>
<i>replace_column</i>	<p>Specifies the column name in the <i>sr_table_spec</i> table or file containing the replacement values. If <i>sr_table_spec</i> is an internal table, this should be set to NULL.</p> <p>All data contained in this column must be of the varchar type.</p>

<i>sr_type</i>	<p>A string that specifies the type of search and replace operation to perform. It has three possible values:</p> <p>'SR_FIELD' — Matches the entire contents of the search field, and replaces the entire contents of the search field.</p> <p>'SR_WORD' — 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>'SR_STRING' — 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>
<i>case_sensitivity</i>	<p>A value that indicates whether or not the search and replace operation is case sensitive.</p> <p>yes — case sensitive</p> <p>no — not case sensitive</p>
<i>default_replace_value</i>	<p>A varchar that specifies the default replacement value if the search value is not found.</p> <p>This applies only when <i>sr_type</i> is set to 'SR_FIELD'.</p>
<i>input_column_list</i>	A varchar that specifies a comma-separated list of input expressions on which the search and replace operation should be performed.
<i>output_column_list</i>	A varchar that specifies a comma-separated list of output columns.
<i>output_variable_list</i>	<p>A varchar that specifies a comma-separated list of output variables.</p> <p>When more than one output column is specified in <i>output_column_list</i>, 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 <i>output_column_list</i>.</p>

SET options	<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"= '&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;searchTable&gt;   &lt;item&gt;     &lt;Search&gt;value&lt;/Search&gt;     &lt;Replace&gt;value&lt;/Replace&gt;   &lt;/item&gt;   ... &lt;/searchTable&gt; ')</pre> <p>For custom SQL:</p> <pre>SET ( "external_custom_sql"= '&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;database_datastore&gt;   datastore_name &lt;/database_datastore&gt; &lt;search_column&gt;   search_column_name &lt;/search_column&gt; &lt;replace_column&gt;   replace_column_name &lt;/replace_column&gt; &lt;SQLText&gt;   custom_sql_string &lt;/SQLText&gt; ')</pre>
-------------	--

**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.

### 6.3.119.1 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
<b>Full string</b>	Matches and replaces the entire contents of the input expression.
<b>Word</b>	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.
<b>Substring</b>	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 Topics**

- [Performance Optimization Guide: Distributing Data Flow Execution, Splitting a data flow into sub data flows, Run as a separate process option](#)

### 6.3.120 `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**

<i>datastore</i>	The name of the CDC-enabled datastore containing the tables used to obtain the changes.
------------------	---

**Example:**

```
set_cdc_checkpoint('MyCdcSource');
```

**6.3.121 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**

<i>variable_name</i>	The name of an environment variable. The name must be surrounded by single quotes.
<i>variable_value</i>	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
<pre>set_env('TMP','C:\Data Services\Temp')</pre>	Sets the value of the TMP environment variable to "C:\Data Services\Temp" and returns a 1.

**Related Topics**

- [get\\_env](#)

**6.3.122 sleep**

Suspends the execution of the calling data flow or work flow.

**Syntax**

```
sleep(num_millisecs)
```

**Return Value**

int

Returns '1', always.

**Where**

<i>num_millisecs</i>	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
```

**6.3.123 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**

<code>input_str</code>	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
<pre>Print(soundex('Hello'));</pre>	Prints the soundex of the word "Hello."
<pre>\$VAR=soundex(emp_name);</pre>	Returns the soundex encoding for the string stored in the variable emp_name and then assigns it to \$VAR
<pre>\$VAR=soundex('1234567');</pre>	Returns '0000' because the input data is numeric.

---

## 6.3.124 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**

<i>datastore</i>	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.
<i>sql_command</i>	<p>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 (\).</p> <p>Data Services makes column and table names uppercase when sending the <i>sql_command</i> to Oracle to resolve. To specify a lowercase column or table name from an Oracle database, enclose the name with double quotation marks (").</p>

**Example:**

Function	Results
<pre>sql('source_ds', 'SELECT EmpID FROM Emp WHERE sal &gt; 100000')</pre>	Runs the SQL command against the database connected through the datastore <code>source_ds</code> .
<pre>sql('source_ds','SELECT customer.LastName FROM customer WHERE customer.State = \'CA\')</pre>	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 <code>sql_command</code> string.
<pre>sql('oracle_ds', ('SELECT "start_timestamp" FROM "status_table" WHERE "extract_name" = \'DF_RecoverDim\' AND "stop_timestamp" = NULL'))</pre>	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.
<pre>\$start_date=sql('warehouse_ds', 'SELECT finish_timestamp FROM time_table WHERE table_name= "Component_Orders" ');</pre>	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.
<pre>\$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');</pre>	This script example assumes the database returns the date in <code>dd-mon-yyyy</code> 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 <code>to_char</code> function in Oracle or the <code>convert</code> function in MS SQL.

**6.3.125 sqrt**

Returns the square root of the given expression.

**Syntax**

```
sqrt(num)
```

**Return Value**

Float

**Where**

<i>num</i>	The number for which you want the square root.
------------	--

**Details**

Return value is NULL if the input is negative.

**Example:**

Function	Results
<code>sqrt(625.25);</code>	25.005000

**6.3.126 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**

<i>recipients_list</i>	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.
<i>subject</i>	A string containing the subject of the e-mail. This string can be empty.
<i>message</i>	A string containing the e-mail message. This string can be empty.

<code>number_of_trace_lines</code>	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.
<code>number_of_error_lines</code>	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
<pre>\$myvar = smtp_to('admin@company.com', 'Out of memory error in the SalesFact job. Please fix the error before running recovery job.', ' ', 10, 10)</pre>	The message goes to one recipient, admin@company.com.
<pre>\$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)</pre>	<p>The message goes to two recipients, admin@company.com and manager@company.com.</p> <p>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.</p> <p>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 <b>Functions</b> button to construct a <code>smtp_to</code> statement.</p>



Function	Results
<pre> \$address_list = 'nkumar@businessob jects.com,pkulkarn@businessobjects.com';  \$subject_text = 'Testing SMTP job smtp011';  \$message_text = 'The job smtp011 has the following trace lines &amp; errors';  \$trace_log = 8;  \$error_log = 19;  \$smtp_output = 9;  print ('before execution :- '    \$smtp_output);  \$smtp_output = smtp_to(\$ad dress_list,\$subject_text,\$message_text, \$trace_log,\$error_log);  print ('after execution  :- '    \$smtp_output); </pre>	<p>The smtp_to function also supports global variables. This example from a script shows the smtp_to function substituting the values from global variables.</p>
<p>Create a custom function my_smtp that contains the following:</p> <pre> begin return(smtp_to('nkumar@ness-gsg.com','Testing job smtp012','Test message of smtp012 job',4,5)); End </pre> <p>Use the custom function my_smtp in a script as follows:</p> <pre> print('before smtp :- '); my_smtp( ); print('after smtp :- '); </pre>	<p>You can use smtp_to in any user-defined custom function and invoke it in a script.</p>

**Note:**

The smtp\_to function does not support nicknames.

### 6.3.126.1 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 BusinessObjects Data Services 4.1 > 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**.

### 6.3.127 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**

<i>input_string</i>	The string to be modified.
<i>start</i>	<p>The position of the first character in the new string. The function counts characters from the beginning of <i>input_string</i>. In normal data flows, the first character is position number 1. If <i>start</i> is 0, the new string begins with the first character (position 1).</p> <p>If <i>start</i> is negative, the function counts characters from the end of <i>input_string</i>. The new string begins with the character in that position from the end of the string.</p> <p>If <i>start</i> is greater than the number of characters in <i>input_string</i>, the function returns an empty string.</p>
<i>length</i>	<p>The number of characters in the resulting string. If <i>length</i> is 0 or negative, the function returns an empty string. If <i>length</i> is greater than the number of characters remaining in <i>input_string</i> after <i>start</i>, the function returns only the remaining characters.</p> <p>The function keeps the trailing blanks in the remaining <i>input_string</i> after <i>start</i>.</p>

**Example:**

Function	Results
<code>substr('94025-3373', 1, 5)</code>	'94025'
<code>substr('94025-3373', 7, 4)</code>	'3373'
<code>substr('94025', 7, 4)</code>	NULL
<code>substr('Dr. Schultz', 4, 18)</code>	'Schultz'
<code>substr('San Francisco, CA', -4, 18)</code>	', CA'

**6.3.128 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**

<i>value_list</i>	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.

### 6.3.129 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
<code>isweekend(sysdate())</code>	Tests whether today is a Saturday or Sunday.
<code>to_char(sysdate(), 'yyyy.mm.dd')</code>	<p>Converts the sysdate function's <code>datetime</code> value to a string that displays only the date.</p> <p>For example, you can use this to exclude part of the <code>datetime</code> data by only providing a format for the data you want to display in a report. To convert a <code>datetime</code> value to a string containing only the date, use this expression and change the column's data type to <code>varchar</code>.</p>

### 6.3.130 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()]');
```

---

### 6.3.131 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
<pre>\$timestamp = sql('my_datastore', ('UPDATE status_table SET job_start_time = \' '    to_char(sys- time(), 'hh24:mi:ss.ff')    '\');</pre>	<p>This expression updates the <code>job_start_time</code> column of the <code>status_table</code> with the current time. It also formats the time data.</p>
<pre>to_char(sys- time(), 'hh24:mi:ss.ff')</pre>	<p>Trims date data from the <code>sys- time()</code> function in cases where it is added by default. Set the column that contains this expression to the data type <code>varchar</code>.</p> <p>The data type for a column that calls the <code>sys- time()</code> function should be <code>time</code>. If the data type is set to <code>datetime</code>, Data Services will add the default date for the <code>datetime</code> data type (1900:01:01) because <code>sys- time()</code> does not read dates.</p>

**6.3.132 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**

<i>table_name</i>	<p>Use the following format: <code>'datastore.owner.table'</code>. If a valid table by this name does not exist, NULL is returned. This parameter is case sensitive.</p>
-------------------	--

<i>attribute_name</i>	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
<code>table_attribute('mssql.avez.CUSTOMER', 'Number_Of_Inserts')</code>	'1788'

**6.3.133 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*.

<i>numeric_expression</i>	The source int, real, double or decimal value.	
<i>format</i>	A string indicating the format of the generated string. 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.	<code>to_char(123,'9999') = ' 123'</code>
0	Number  Including leading/trailing zeros.	<code>to_char(123,'09999') = ' 0123'</code> <code>to_char(123,'9999D.00') = ' 123.00'</code>
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.	<code>to_char(12.34,'99D.99') = ' 12.34'</code>
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.	<code>to_char(1234,' 9G,999') = ' 1,234'</code>
x	String containing unsigned hexadecimal integer, using "abcdef". Output is not padded if number is not 2 bytes long.	<code>to_char(123,'xx') = ' 7b'</code> <code>to_char(12,'x') = ' c'</code>
X	String containing unsigned hexadecimal integer, using "ABCDEF". Output is not padded if number is not 2 byte long.	<code>to_char(123,'XX') = ' 7B'</code> <code>to_char(12,'X') = ' C'</code>
0	String containing unsigned octal integer. This option is "not" case sensitive. Output is not padded if number is not 2 bytes long.	<code>to_char(12,'oo') = ' 14'</code> <code>to_char(1,'o') = ' 1'</code>

A formatted string describing *date*.

<i>date</i>	The source date, time, or datetime value.	
<i>format</i>	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
<pre>to_char(call_date, 'dd-mon-yy hh24:mi:ss.ff')</pre>	<p>The date value from the <code>call_date</code> column formatted as a string, such as:</p> <pre>28-FEB-97 13:45:23.32</pre>

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 Topics

- [timestamp](#)

### 6.3.134 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

<i>input_string</i>	The source string.	
<i>format</i>	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 (0-23)
	MI	2-digit minute (0-59)
	SS	2-digit second (0-59)
	FF	Up to 9-digit sub-seconds

**Example:**

Function	Results
<code>to_date('Jan 8, 1968', 'mon dd, yyyy')</code>	1968.01.08 stored as a date.

**Related Topics**

- [timestamp](#)

**6.3.135 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**

<i>in_str</i>	The number string. Null implies a NULL return.
<i>decimal_sep</i>	The character that separates the decimal component from the whole number component.
<i>thousand_sep</i>	The character that separates thousands from hundreds in the whole number component.
<i>scale</i>	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
<code>to_decimal('99,567.99', '.', ',', 3)</code>	99567.990

**6.3.136 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**

<i>in_str</i>	The number string. Null implies a NULL return.
<i>decimal_sep</i>	The character that separates the decimal component from the whole number component.
<i>thousand_sep</i>	The character that separates thousands from hundreds in the whole number component.
<i>precision</i>	The total number of digits in the returned value.
<i>scale</i>	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
<code>to_decimal_ext('99,567.99', '.', ',', ' ', 38, 3)</code>	99567.990

---

### 6.3.137 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**

<i>datastore</i>	The name of the datastore containing the table.
<i>owner</i>	The name of the datastore owner. Not used for memory tables.
<i>table_name</i>	The name of the database table or memory table containing the rows you want to count.

**Example:**

Function	Results
<code>total_rows (ora_ds.scott.emp_table)</code>	Retrieves the total number of rows from an Oracle table.
<code>total_rows (mem_ds..bigtable)</code>	Retrieves the total number of rows from a memory table.

**6.3.138 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**

<i>num1</i>	The source number.
<i>precision</i>	An integer indicating the number of decimals in the result. If <i>precision</i> is negative, digits to the left of the decimal point are truncated and the value is padded with zeros.

**Example:**

Function	Results
<code>trunc(120.12345, 2)</code>	120.12
<code>trunc(120.12999, 2)</code>	120.12
<code>trunc(180, -2)</code>	100
<code>trunc(120.123, 5)</code>	120.12300

**6.3.139 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**

<i>ds</i>	The datastore containing the memory table or persistent cache table.
<i>tab_name</i>	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
<code>truncate_table (ds..bigtable)</code>	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.

**6.3.140 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**

<i>value</i>	The string to be modified.
<i>locale</i>	Optional parameter that converts the string to the specified locale. <b>Note:</b> ISO 639 language code and ISO 3166 country code formats are supported.

**Example:**

Function	Results
<code>upper('Accounting101')</code>	'ACCOUNTING101'
<code>upper(substr(LastName,1,1))   lower(substr(LastName,2,LENGTH(LastName)))</code>	The value in column <code>LastName</code> with the first letter uppercase and the rest of the value lowercase. Note that this example does not account for two-word last names.
<code>upper(LastName, 'tr')</code>	The value in column <code>LastName</code> is converted to all lowercase. It is also converted to the Turkish locale, using the ISO 639 language code.

**Related Topics**

- [ISO 639 language list](#)
- [ISO 3166 Country Code list](#)

**6.3.141 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**

<i>column_name</i>	The name of the table column for which you want to convert a data type from varchar to long.
--------------------	--

**Related Topics**

- [Designer Guide: XML extraction and parsing for columns, Scenario 2](#)

### 6.3.142 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

<i>file_name_pattern</i>	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.
<i>timeout</i>	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.
<i>poll_interval</i>	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.
<i>max_match</i>	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.
<i>file_name_list</i>	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.
<i>list_size</i>	Optional. Output integer variable that returns the list size.

<code>list_separator</code>	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.

---

### 6.3.143 `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**

<i>date1</i>	The source date.
--------------	------------------

**Example:**

Function	Results
<code>week_in_month(to_date('Jan 22, 1997', 'mon dd, yyyy'))</code>	4
<code>week_in_month(to_date('Jan 21, 1997', 'mon dd, yyyy'))</code>	3

**6.3.144 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**

<i>inputdate</i>	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
<code>week_in_year(to_date('Jan 01, 2001', 'mon dd, yyyy'))</code>	1
<code>week_in_year(to_date('2005.01.01', 'yyyy.mm.dd'), 'WW')</code>	1
<code>week_in_year(to_date('2005.01.01', 'yyyy.mm.dd'), 'IW')</code>	53

### 6.3.145 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'.
```

### 6.3.146 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**

<i>input_string</i>	The source string.
<i>word_num</i>	A nonnegative integer specifying the index of the target word in the string. The first word in a string is word number 1. If <i>word_num</i> is 0 or greater than the number of words in <i>input_string</i> , 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
<code>word('Accounting Department', 1)</code>	'Accounting'
<code>word('Accounting', 1)</code>	'Accounting'
<code>word('Accounting', 2)</code>	NULL

---

**6.3.147 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**

<i>string</i>	The source string.
<i>word_num</i>	A nonnegative integer specifying the index of the target word in the string. The first word in a string is word number 1. If <i>word_num</i> is 0 or greater than the number of words in <i>string</i> , then the word function returns a NULL string.
<i>separator(s)</i>	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
<code>word_ext('www.sap.com', 2, '.')</code>	'sap'
<code>word_ext('www.cs.wisc.edu', -2, '.')</code>	'wisc' A negative word number means count from right to left.
<code>word_ext('www.cs.wisc.edu', 5, '.')</code>	NULL
<code>word_ext('aaa+=bbb+=ccc+zz=dd', 4, '+=')</code>	'zz' If 2 separators are specified (+=), the function looks for either one.
<code>word_ext(',,,,,aaa,,,,bb,,,c ', 2, ',')</code>	'bb' This function skips consecutive delimiters.

**6.3.148 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()]')
```

### 6.3.149 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

<i>date1</i>	The source date.
--------------	------------------

#### Example:

Function	Results
<code>year(to_date('Jan 22, 1997', 'mon dd, yyyy'))</code>	1997
<code>year(to_date('03/97', 'mm/yy'))</code>	1997
<code>year(to_date('03/19', 'mm/yy'))</code>	2019

## 6.4 Custom functions

You can create your own functions by writing script functions in SAP BusinessObjects 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 Topics**

- [Scripting Language](#)

### **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 Topics

- [Smart editor](#)

## 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 Topics**

- [Custom functions](#)
- [To edit an existing function call in an expression](#)

### 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.

### 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.

## 6.5 About procedures

Data Services supports the use of stored procedures for DB2, ODBC, Oracle, Microsoft SQL Server, SAP HANA, Sybase ASE, Sybase IQ, and Teradata databases. You can call stored procedures from the jobs you create and run in Data Services.

### 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, Sybase ASE, 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.

## 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`.

### **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.

### 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 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

### 6.5.3.1 Creating stored procedure 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 Topics

- [Creating stored procedures in a database](#)



### 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 Topics**

- [Creating stored procedures in a database](#)

### 6.5.3.3 Creating stored procedures in SAP HANA

SAP BusinessObjects 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 Topics**

- [Creating stored procedures in a database](#)

### 6.5.3.4 Creating stored procedures in MS SQL Server or Sybase ASE

In Microsoft SQL Server and 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 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 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 Topics

- [Creating stored procedures in a database](#)

## 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 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`.

**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 Topics**

- [Checking execution status](#)
- [Designer Guide: Datastores, Ways of importing metadata](#)

## 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)

**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 Topics**

- [Checking execution status](#)
- [Data Types](#)

## 6.5.6 Calling stored procedures

This section discusses calling stored procedures.

### 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 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 Topics

- [From queries](#)
- [To include a stored procedure in a query output schema](#)

## 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 BusinessObjects 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 Topics

- [In general](#)
- [Importing metadata for stored procedures](#)

#### 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 BusinessObjects 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`.



### 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 Topics

- [Checking execution status](#)

## 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.

# Scripting Language

This section describes the SAP BusinessObjects 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.

## 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 Topics

- [Language syntax](#)
- [Sample scripts](#)

## 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.

### 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 Topics

- [Sample scripts](#)

## 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.

## 7.2.3 Strings

This section contains the following information about strings:

- Quotation marks
- Escape characters
- Trailing blanks

### Related Topics

- [Quotation marks](#)
- [Escape characters](#)
- [Trailing blanks](#)
- [NULL values and empty strings](#)

### 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: <ul style="list-style-type: none"><li>• String constants (for example, 'Hello World' or '1995.01.23')</li><li>• Numeric constants (for example, 2.14)</li></ul>

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.

### 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'

### 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.

## 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 Topics

- [Designer Guide: Variables and Parameters](#)
- [To create a custom function](#)

## 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 (\).

## 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 editor under the datastore used to import them.

**Related Topics**

- [To create a custom function](#)
- [Importing metadata for stored procedures](#)

## 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
```

## 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 Topics

- [nvl](#)

### 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 NULLs 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
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
<code>if(\$var1 = \$var2)</code>	Do not compare without explicitly testing for NULLs. Using this logic is not recommended because any relational comparison to a NULL value returns FALSE.
<code>if ( ((\$var1 IS NULL) AND (\$var2 IS NULL)) OR (\$var1 = \$var2) )</code>	Will execute the TRUE branch if both \$var1 and \$var2 are NULL, or if neither are NULL but are equal to each other.

## 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:

<b>Section:</b>	Enter "Parser".
<b>Key:</b>	Enter "NumErrors".
<b>Value:</b>	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.

## 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 Topics

- [About Reserved Words](#)

### 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.

### 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
```

### 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.

### 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.

### 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 Topics**

- [RepeatString function](#)

### **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.

### **7.2.10.7 TRY**

Indicates the start of a try/catch block.

#### **Related Topics**

- [CATCH](#)

### **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.

## 7.3 Sample scripts

The following examples show how a script would be formatted for the Square and Repeat String functions.

### 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);
```

### 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	



Description	Value	Arguments
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
```



## 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*.

### 8.1 Auditing metadata

This set of tables in the Data Services repository stores the statistics that the audit feature collects.

- [AL\\_AUDIT](#)
- [AL\\_AUDIT\\_INFO](#)

#### 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.

Column Name	Data Type	Description
STATUS	INTEGER	Audit status can be one of the following values: <ul style="list-style-type: none"> <li>• 0 — Not audited</li> <li>• 1 — Audit rule succeeded</li> <li>• 2 — Audit information collected. This status occurs when you define audit labels to collect statistics but do not define audit rules.</li> <li>• 3 — Audit rule failed</li> </ul>
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
```

**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: <ul style="list-style-type: none"> <li>• Number of rows in a table or whole row set</li> <li>• Sum of the values in a column</li> <li>• Average of the values in a column</li> <li>• Checksum of the values in a column</li> </ul>

**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';
```

---

## 8.2 Imported metadata

This set of tables (prefixed by AL\_) and views (prefixed by ALVW\_) capture information about the metadata imported into SAP BusinessObjects Data Services from external databases and applications (such as Oracle, PeopleSoft, and SAP Applications).

- [AL\\_INDEX](#)
- [AL\\_PCOLUMN](#)
- [AL\\_PKEY](#)
- [ALVW\\_COLUMNATTR](#)
- [ALVW\\_COLUMNINFO](#)
- [ALVW\\_FKREL](#)
- [ALVW\\_MAPPING](#)
- [ALVW\\_TABLEATTR](#)
- [ALVW\\_TABLEINFO](#)

### 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'
```

## 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.

### 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.
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'
```

### 8.2.4 ALVW\_COLUMNATTR

This view contains information about column attributes.

Column Name	Data type	Description
TABLE_NAME	Varchar(64)	Name of the table.
TABLE_OWNER	Varchar(64)	Owner of the table.

Column Name	Data type	Description
DATASTORE	Varchar(64)	The datastore to which this table belongs.
COLUMN_NAME	Varchar(64)	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'
```

**8.2.5 ALVW\_COLUMNINFO**

This view contains information about the columns in a table.

Column Name	Data type	Description
TABLE_NAME	Varchar(64)	Name of the table.
TABLE_OWNER	Varchar(64)	Owner of the table.



Column Name	Data type	Description
DATASTORE	Varchar(64)	The datastore to which this table belongs.
COLUMN_NAME	Varchar(64)	Name of the column.
COLUMN_DATATYPE	Varchar(20)	Data type of this column. Examples: integer, date time, 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'
```

## 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'
```

## 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.

Table 8-9: ALVW\_MAPPING view

Column Name	Data type	Description
DF_NAME	varchar(64)	Data flow that populates the target table.
TRG_TAB_NAME	varchar(64)	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(64)	Owner of the target table.
TRG_DS	varchar(64)	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(65)	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(64)	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(64)	Owner of the source table.
SRC_DS	varchar(64)	Datastore of the source table.
SRC_TYPE	varchar(64)	Type of source. Examples: table, file.
SRC_COL_NAME	varchar(65)	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 Topics

- [Storing nested column-mapping data](#)

### 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'
```

### 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	<p>The target column is mapped directly from a source column with no expression to transform it.</p> <p>For example, <code>EMPID</code> (employee ID) mapped directly from source to target.</p>
Computed	<p>There is an expression associated with the target column.</p> <p>For example, <code>NAME</code> is <code>LAST_NAME    ', '    FIRST_NAME</code>.</p>

Mapping Type	Description
Generated	<p>There is no source column associated with the target column.</p> <p>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.</p>
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.

### 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	<code>((Price x 1.17) x 112)</code>

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.

### 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.

### 8.2.7.5 Storing nested column-mapping data

SAP BusinessObjects 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 (65) and not big enough to store long NRDM column names, the software uses the AL\_COLMAP\_NAMES table to support nested data.

Table 8-12: AL\_COLMAP\_NAMES table

Column Name	Data type	Description
DF_NAME	varchar(64)	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(255)	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 255 characters is required to store data in the COL_NAME column. For each set of 255 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



ALVW_MAPPING view	Column value
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
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.

### 8.2.8 ALVW\_TABLEATTR

This view contains information about the attributes in a table.

Column Name	Data type	Description
TABLE_NAME	Varchar(64)	Name of the table.

Column Name	Data type	Description
TABLE_OWNER	Varchar(64)	Owner of the table. For SAP applications, the value is NULL.
DATASTORE	Varchar(64)	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'
```

## 8.2.9 ALVW\_TABLEINFO

This view contains the list of tables imported into the repository.

Column Name	Data type	Description
TABLE_NAME	Varchar(64)	Name of the table.
TABLE_OWNER	Varchar(64)	Owner of the table. For SAP applications, this table is NULL.

Column Name	Data type	Description
DATASTORE	Varchar(64)	The datastore to which this table belongs.
TABLE_ID	int	Internal ID of the table within the repository.

## 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](#)
- [AL\\_LANGXMLTEXT](#)
- [AL\\_ATTR](#)
- [AL\\_USAGE](#)
- [ALVW\\_FUNCINFO](#)
- [ALVW\\_PARENT\\_CHILD](#)

### 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.

Column Name	Description
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)
```

### 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.

Column Name	Description
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.
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.

### 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)
```

### 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.

Column Name	Data type	Description
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.

### 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](#).

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.  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.

Column Name	Data type	Description
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(64)	Owner of the child table.
DEPTH	int	Indicates the number of levels between objects in a parent/descendent relationship.

### 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	Dataflow	SALES_ORDER	Table	Sales order target fact table	Target
Daily_Job	Workflow	SALES_ORDER	Table	Sales order target fact table	Target
DF_NewOrders	Job	SALES_ORDER	Table	Sales order target fact table	Target
Get_IDoc	Dataflow	SALES_ORDER	Table	Sales order target fact table	Target
IDoc_job	Dataflow	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.DECEN_OBJ,
AL_USAGE.DECEN_OBJ_DESC,
AL_USAGE.DECEN_OBJ_TYPE,
AL_USAGE.DECEN_OBJ_USAGE
FROM AL_USAGE
WHERE (AL_USAGE.DECEN_OBJ_TYPE = 'Table'
AND AL_USAGE.DECEN_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	Dataflow	SALES_ORDER	Table	Sales order target fact table	Target
Daily_Job	Workflow	SALES_ORDER	Table	Sales order target fact table	Target
DF_NewOrders	Job	SALES_ORDER	Table	Sales order target fact table	Target
Get_IDoc	Dataflow	SALES_ORDER	Table	Sales order target fact table	Target
IDoc_job	Dataflow	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.DSCEN_OBJ,
  AL_USAGE.DSCEN_OBJ_DESC,
  AL_USAGE.DSCEN_OBJ_TYPE,
  AL_USAGE.DSCEN_OBJ_USAGE
FROM AL_USAGE
WHERE (AL_USAGE.PARENT_OBJ_TYPE = 'Job'
  AND AL_USAGE.DSCEN_OBJ_TYPE = 'Table'
  AND AL_USAGE.DSCEN_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

### 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.
DATASTORE	Applies only to imported functions. Datastore to which the function belongs.

### 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'
```

---

## 8.4 Operational metadata

These tables and views contain information about the run-time statistics of Data Services jobs and data flows.

- [AL\\_HISTORY](#)
- [ALVW\\_FLOW\\_STAT](#)

### 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).

Column Name	Description
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)
```

## 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.

## Data Quality repository statistics tables

### 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.

**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

Report	Repository table	Transform(s)
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 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





Report	Repository table	Transform(s)
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 Topics**

- [Match repository statistics tables](#)

**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.

### 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
ADDRINFOCODE DATA	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
ADDRIN FOCODESUMMARY	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
ADDRSTATUS CODEDATA	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

Repository table name	Description	Transform
ADDRVALIDATE SUMMARY	Contains record validation statistics found by the transform. Used for the Address Validation Summary.	Global Address Cleanse USA Regulatory Address Cleanse
AMASADRPROC SUMMARY	Contains summary statistics for address processing. Used for the AMAS (Australia Matching Approval System) Address Matching Processing Summary.	Global Address Cleanse
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
DSFSEQUENCES TATS	Contains DSF2 sequence statistics per post-code/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

Repository table name	Description	Transform
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. Used for the SendRight New Zealand Statement of Accuracy (SOA) report.	Global Address Cleanse
USREGULATORYLOCKING	Contains information about the record that caused the lock. Used for the US Regulatory Locking Report.	USA Regulatory Address Cleanse

### 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 Topics**

- [Repository tables common columns](#)

### 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)	<p>Name of the engine applicable to the listed INFOCODE.</p> <p>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).</p> <p>For the USA Regulatory Address Cleanse transform, it is always USA.</p>
 DATA_SOURCE_ID	VARCHAR (80)	Code that uniquely identifies the list that contains the listed INFOCODE.

**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 Topics**

- [Repository tables common columns](#)

### 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.
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 is not applicable to this table.



**Related Topics**

- [Repository tables common columns](#)

**9.3.4 ADDRYPESUMMARY**

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
UNIQSUBURBAN-RECS	INT	Number of records that were determined to be unique suburban addresses.
MOBILER-OUTERECS	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.
UNDERTERMIN-DRECS	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 Topics**

- [Repository tables common columns](#)

**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.
 COMPONENT-NAME	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.
 SUITE_LINK_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 Topics**

- [Repository tables common columns](#)

### 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)
PROCESSING-DATE	DATETIME	Date and time the job was processed through the software in YYYY-MM-DD HR:MIN:SEC format.
RE-SERVED_COUNT1	INT	Reserved for future record counts.
RE-SERVED_COUNT2	INT	Reserved for future record counts.
RE-SERVED_COUNT3	INT	Reserved for future record counts.
RE-SERVED_STRING1	NVARCHAR (100)	Reserved for future strings.
RE-SERVED_STRING2	NVARCHAR (100)	Reserved for future strings.
RE-SERVED_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 Topics**

- [Repository tables common columns](#)

**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
 SUITE_LINK_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)
SUITE_LINK_MATCH_A	INT	Number of records that matched to SuiteLink and had secondary information added.
SUITE_LINK_NO-MATCH_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 Topics**

- [Repository tables common columns](#)

**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
 LI-CENSEE_NUMBER	NVARCHAR (4)	DSF2 licensee number.
 SEQNUM	INT	Sequence number from 0000 to 9999. <ul style="list-style-type: none"> <li>• If the Postcode2 field is blank, this column is blank.</li> <li>• If the Postcode2 field is not blank and the DPV_Status is not Y, then the column contains "0000".</li> </ul>
LI-CENSEE_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.
 DA-TA_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 Topics**

- [Repository tables common columns](#)

**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.
 SORT-CODE_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 Topics**

- [Repository tables common columns](#)

## 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.
 DA-TA_SOURCE_ID	NVARCHAR (80)	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 Topics**

- [Repository tables common columns](#)

### 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: <ul style="list-style-type: none"> <li>• USPS-assigned license ID (first 4 characters)</li> <li>• List Owner NAICS Code (next 5 characters)</li> <li>• Frequency of processing (next 2 characters)</li> <li>• Licensee-assigned List ID (last 6 characters)</li> </ul>
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 occurred. 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 occurred. 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). B (Yes, data modifications from postal and other sources).



Column	Data type definition	Description												
CONCURRENT_PROCESSES	VARCHAR (1)	Indicates that the list was concurrently processed and the data modifications that occurred. 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).												
LIST_NAME	VARCHAR (30)	Name of the list being processed.												
LICENSEE_ID	VARCHAR (4)	Unique ID for the broker or list administrator.												
LICENSEE_COMPANY	VARCHAR (50)	Broker or list administrator name.												
PROCESSING_CATEGORY	VARCHAR (20)	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>	Option	Printed on report	Marketing	MKTG TEST	Normal	NORMAL	Stage I	STAGE I	Stage II	STAGE II	System testing	SYS TEST
Option	Printed on report													
Marketing	MKTG TEST													
Normal	NORMAL													
Stage I	STAGE I													
Stage II	STAGE II													
System testing	SYS TEST													
SERVICE_PROVIDER	VARCHAR (1)	NCOALink list processor's provider level. Values are:  0 End user  1 Limited service provider  2 Full service provider												
NCOAL_PROCESS_DATE	VARCHAR (10)	Date based on the time that the list was processed. Automatically generated by the software, and included in reports.												
LIST_RETURN_DATE	VARCHAR (10)	Date obtained from the <b>List returned date</b> option in the USPS License Information group.												
DPV_ENABLED	VARCHAR (1)	DPV is enabled in the transform (Y/N indicator in reports).												

Column	Data type definition	Description
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.)
PROCESS_STD	VARCHAR (1)	Standard mail was processed by the transform. (Y/N indicator in reports.)

Column	Data type definition	Description
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.

Column	Data type definition	Description
RE-SERVED_STRING2	VARCHAR (100)	Extra field for additional string.
RE-SERVED_STRING3	VARCHAR (100)	Extra field for additional string.
 DA-TA_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 Topics**

- [Repository tables common columns](#)

**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).

Column	Data type definition	Description
AD-DRESS_SOURCE	NVARCHAR (1)	D = Derived by data S = Derived from software
DETAILED_DESC	VARCHAR (1)	Provides a more detailed description of the listed code.
 DA-TA_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 Topics**

- [Repository tables common columns](#)

**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
 LICENSEID	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.

Column	Data type definition	Description
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.

**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 Topics**

- [Repository tables common columns](#)

**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
CASSCOMPANY-NAME	NVARCHAR (50)	Name of the company that is CASS-certified.
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.
Z4COMPANY-NAME	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.
eLOTCOMPANY-NAME	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.

Column	Data type definition	Description
MASSCOMPANY-NAME	NVARCHAR (40)	Not populated from the software.
MASSCONFIGURATION	NVARCHAR (10)	Not populated from the software.
MASSSOFTWARE	NVARCHAR (60)	Not populated from the software.
MLOCRSERIAL-NUMBER	NVARCHAR (20)	Not populated from the software.
LISTPROCESSOR	NVARCHAR (35)	MASS list processor. Not populated from the software.
ZIP4DBDATE	VARCHAR (15)	ZIP4 directory date.
Z4CHANGE-DATE	VARCHAR (15)	Not populated from the software.
Z4CHANGEDB-DATE	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.
ADDRESSLIST-NAME	NVARCHAR (20)	Name of the address list processed. (Entered in <b>CASS Report Options &gt; List Name.</b> )
NUM-BEROFLISTS	INT	Not populated from the software.
TOTALRECPROCESSED	INT	Number of records processed in the job.
ZIP4RECORD-SCODED	INT	Number of ZIP+4 records that were DPV confirmed.
Z4CHGRECORD-SCODED	INT	Number of records that were Z4Change coded.
DPBRECORD-SCODED	INT	Number of records with delivery point bar codes.
DPBCDATE-FROM	VARCHAR (15)	Not populated from the software.
FIVEDIGITREC-CODED	INT	Number of records that were assigned 5-digit ZIP Codes.

Column	Data type definition	Description
FIVEDIGITDATE-FROM	VARCHAR (15)	Not populated from the software.
CRRTRECORD-SCODED	INT	Number of records that were assigned carrier route codes.
CRRDATEFROM	VARCHAR (15)	Not populated from the software.
eLOTRECORD-SCODED	INT	Number of records that were assigned eLOT codes.
eLOTDATE-FROM	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.
HIGHRISEDE-FAULT	INT	Number of records that were assigned as highrise defaults.
HIGHRISEEX-ACT	INT	Number of records that were assigned as highrise exact matches.
RURALROUTE-EDEFAULT	INT	Number of records that were assigned as rural route default matches.
RURALROUTE-EXACT	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.
RDI	INT	Not populated from the software (always zero).
RE-SERVED_COUNT1	INT	Reserved for future record counts.



Column	Data type definition	Description
RE-SERVED_COUNT2	INT	Reserved for future record counts.
RE-SERVED_COUNT3	INT	Reserved for future record counts.
RE-SERVED_STRING1	NVARCHAR (100)	Reserved for future strings.
RE-SERVED_STRING2	NVARCHAR (100)	Reserved for future strings.
RE-SERVED_STRING3	NVARCHAR (100)	Reserved for future strings.
 DA-TA_SOURCE_ID	NVARCHAR (80)	Unique identification 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 Topics**

- [Repository tables common columns](#)

### 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.
CERTPRODUCTNAME	VARCHAR 40	Name of the product that is SendRight certified. This value will be Data Services.
CERTPRODUCT-VER	VARCHAR 20	Version of the SendRight-certified software.
PAFVERSION	VARCHAR 20	Version of the current (most recent) PAF. 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.
CUSTOMER-NAME	VARCHAR 40	Name of the list owner or list processor.
MAILERADDRESS1-6	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.
TOTAL-RECPROC	INT	Number of records processed.
UNIQUE-MATCHES	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.

Column	Data type definition	Description
ADDRESSACCURACY	VARCHAR 8	Percentage of the total records validated from the total number of records processed.
RE-SERVED_COUNT1-3	INT	Reserved for future use.
RE-SERVED_STRING1-3	VARCHAR 255	Reserved for future use.

### 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.
VENDORADDRESS1-2	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.
SREXPYDATE	VARCHAR (11)	Expiration date of the software's SERP certification.
CUSTOMER-NAME	VARCHAR (40)	Company name of the organization for whom the mailing is being prepared.
CUSTOMERADDRESS1-4	VARCHAR (40)	Name and address of the person or organization for whom the mailing is being prepared. (Four columns, 40 varchars each.)
CUSTOMERCPC- NUM	VARCHAR (15)	Date of the postalcode file.

Column	Data type definition	Description
CPCMASTER-FILEVER	VARCHAR (11)	Customer's CPC number that is located in the Canada Post Contract.
TOTAL-RECPROC	INT	Number of urban and rural records processed.
REASSIGNED	INT	Number of records assigned.
QUEST_RECS_RURAL	INT	Number of questionable rural records.
QUEST_RECS_APPT	INT	Number of questionable apartment records.
RE-SERVED_COUNT1-3	INT	Reserved for future record counts.
RE-SERVED_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 Topics**

- [Repository tables common columns](#)

## 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 postcode from the address that caused the lock.
PRIMARY_NUMBER	VARCHAR (81)	Input primary range from the address that caused the lock.

Column	Data type definition	Description
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.
 DA-TA_SOURCE_ID	NVARCHAR (80)	Unique identifier for 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 Topics**

- [Repository tables common columns](#)

## 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

Repository table name	Description	Transform
GEO_INFO_CODE	Contains Geocoder information codes and record counts per code. Used in the US Addressing Report.	Geocoder

### 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 Topics**

- [Repository tables common columns](#)

- [Output fields](#)

## 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.

**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 Topics**

- [Repository tables common columns](#)
- [Output fields](#)

## 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.
MTGSSRCBYSRC-STS	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).

### 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.  <b>Note:</b> 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.

**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 Topics**



- [Repository tables common columns](#)

## 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.

**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).

Column	Data Type Definition	Description
PROTECT-DROPS	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.

**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 Topics**

- [Repository tables common columns](#)

**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).

**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).
CMPBUF-MAXRECS	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).

Column	Data type definition	Description
BRKGRPCOUNT	INT	Number of break groups formed based on the break group strategy.
BRKGR-PLARGEST	INT	Largest break group processed.
BRKGRPCOM-PARES	FLOAT	Number of comparisons made in all the break groups.

**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 Topics**


- [Repository tables common columns](#)

## 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 Topics**

- [Repository tables common columns](#)

## 9.5.5 MTCMPCRT

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
 CMPCRTID	INT	Sequential number that identifies the criteria of a match level.
NAME	NVARCHAR (256)	Match level name.
WTMTSCORE	INT	Minimum weighted match score needed to make a match decision.
NUM-NAMEMUSTMT	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 <b>Compare Given Name1 to Given Name2</b> option. (Y/N)
MTONHY-PLNAME	CHAR (1)	Specifies the setting for the <b>Match on hyphenated family name</b> option. (Y/N)
TRNONMAIDE-NADJ	CHAR (1)	Specifies the setting for the <b>Ignore family name when female</b> option. (Y/N)
IGNFIRMIFNAME	CHAR (1)	Specifies the setting for the <b>Ignore Firm if Name matches</b> option. (Y/N)
IGNORESTIF-BOX	CHAR (1)	Specifies the setting for the <b>Match on Street and RR, or on Box</b> option. (Y/N)
ADDRBLMIF-FIRM	CHAR (1)	Specifies the setting for the <b>Address matches blank if Firms match</b> option. (Y/N)
UNIQRESR-RNOBOX	CHAR (1)	Specifies the setting for the <b>Unique on resident if RR, but no Box</b> 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 Topics**

- [Repository tables common columns](#)

**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.
ONEFLD-BLNKOP	CHAR (1)	Setting for the One Field Blank operation. Values are E (evaluate) and I (ignore).
ONEFLD-BLNKSCORE	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).
BTFLD-BLNKSCORE	INT	Score given to the criteria when both fields compared are blank.
CN-TRBTOWTSCORE	INT	Weight percentage given to the match criteria's contribution to the weighted score.
USEINWTSCOR-EIFGT	INT	Minimum similarity score needed to qualify the match criteria for use in determining the weighted match score.
ZWTSCOREI-FLTOREQ	INT	Minimum similarity score needed for the match criteria to qualify for contributing a value other than zero to the weighted match score.

Column	Data type definition	Description
CMPALGO	CHAR (1)	String comparison algorithm that was used. Values are F for field or W for word.
CHKTRANSP-SLET	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 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: <ul style="list-style-type: none"> <li>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.</li> <li>Letters that match are given a score of 100. The remaining letters are given the score that you specify.</li> <li>The two scores are proportionally combined to render the overall score.</li> </ul>
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 Topics**

- [Repository tables common columns](#)

## 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.

**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 Topics**

- [Repository tables common columns](#)

## 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.

**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 Topics**

- [Repository tables common columns](#)

**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.

Column	Data type definition	Description
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.

**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 Topics**

- [Repository tables common columns](#)

## 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 Topics**

- [Repository tables common columns](#)

## 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 Topics**

- [Repository tables common columns](#)

**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
 INSRCGRP-ID	INT	Sequential number that identifies an input source group.
INSRCGRP-NAME	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.
DE-FAULT_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 Topics**

- [Repository tables common columns](#)

### 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.
 INSRCGRP-ID	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.
DE-FAULT_COUNT	INT	Number of records that belong to this input source because it is the default input source.

**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 Topics**

- [Repository tables common columns](#)

**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 INSRCGRP-ID must be added.

Column	Data type definition	Description
 PROCID	INT	Sequential number that identifies a match level or an association.
 INSRCID	INT	Sequential number that identifies an input source.
 INSRGRPID	INT	Sequential number that identifies an input source group. When INSRGRPID 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.

**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 Topics**

- [Repository tables common columns](#)

**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.
 INSRCGRP-ID	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	NVARNVAR-CHAR (15)	Name of the input source group statistics object.
SECSRCNAME	NVARNVAR-CHAR (255)	Currently not used.
SSMAS-TERS_SEL	NVARNVAR-CHAR (1)	Indicates whether the <b>Single source masters</b> option is selected (Y/N).
SSMAS-TERS_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.
SSMAS-TERS_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.
MSMAS-TERS_SEL	NVARCHAR (1)	Indicates whether the <b>Multiple source masters</b> option is selected (Y/N).
MSMAS-TERS_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.
MSMAS-TERS_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 <b>Single source subordinates</b> option is selected (Y/N).
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.

Column	Data type definition	Description
SS-SUBS_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 <b>Multiple source subordinates</b> 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 <b>Suppression subordinates</b> option is selected (Y/N).
SUP-SUBS_KEEP	INT	Number of suppress subordinate records in INSRCID that are flagged for selection.
SUP-SUBS_DROP	INT	Number of suppress subordinate records in INSRCID that are not flagged for selection.
UNIQUES_SEL	NVARCHAR (1)	Indicates whether the <b>Unique</b> 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.
SUPMAS-TERS_SEL	NVARCHAR (1)	Indicates whether the <b>Suppression masters</b> option is selected (Y/N).
SUPMAS-TERS_KEEP	INT	Number of suppress master records in INSRCID that are flagged for selection.
SUPMAS-TERS_DROP	INT	Number of suppress master records in INSRCID that are not flagged for selection.
SUPMATCH-ES_SEL	NVARCHAR (1)	Indicates whether the <b>Suppression matches</b> (normal and special records that follow a suppress record in a match group) option is selected (Y/N).
SUPMATCH-ES_KEEP	INT	Number of suppress match records in INSRCID that are flagged for selection.
SUPMATCH-ES_DROP	INT	Number of suppress match records in INSRCID that are not flagged for selection.



Column	Data type definition	Description
SUPUNQUES_SEL	NVARCHAR (1)	Indicates whether the <b>Suppression uniques</b> option is selected (Y/N).
SUPUNQUES_KEEP	INT	Number of unique suppress records in INSRCID that are flagged for selection.
SUPUNQUES_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 Topics**





- [Repository tables common columns](#)

## 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.
 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.

Column	Data type definition	Description
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 Topics**

- [Repository tables common columns](#)

**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.
 KEYFLDALT- NUM	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.
DATARECFLD- NAME	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).
CONVTXTTON- UM	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 Topics**

- [Repository tables common columns](#)

**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 Topics**

- [Repository tables common columns](#)

## 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.

**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 Topics**

- [Repository tables common columns](#)



# 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 Topics

- [Definitions](#)
- [Supported locales and encodings](#)

## 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 generates 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.

### 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.

**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.

## 10.1.2 To set locales in the Locale Selector

Perform this task if you are installing the engine but not the Designer.

**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 BusinessObjects Data Services 4.1 > 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.

## 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.



### 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:

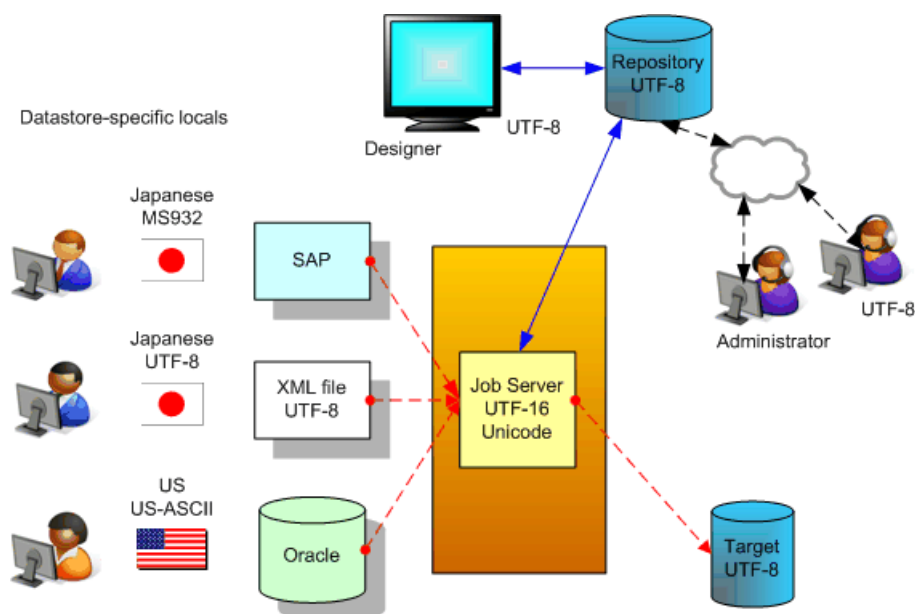
Component	Description
License Manager	License Manager always displays in English.
Management Console	The Management Console's locales are controlled in your internet browser's locale setting.
Documentation	Documentation that is accessed from the Designer will display in the same language as the Designer's "Options" window <b>Product locale</b> setting.
Log files	Messages written to a log file will be in the language set in the <b>Server Logs locale</b> option in the Locale Selector. This affects only the text of the log files. The date and timestamps remain the same.

## 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.

**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.
- 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 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 Topics

- [Processing with and without UTF-16 Unicode](#)
- [File format locales](#)

## 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
Single-byte code page 1	Single-byte code page 2	Single-byte code page 3	Unicode

Datastore 1 locale	Datastore 2 locale	Job Server locale	Data flow locale
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 textual 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.

### 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 BusinessObjects Data Services 4.1 > Data Services Locale Selector**.

**Note:**

For more information, see “Guidelines for setting locales”.

## 10.2.2 Code page support

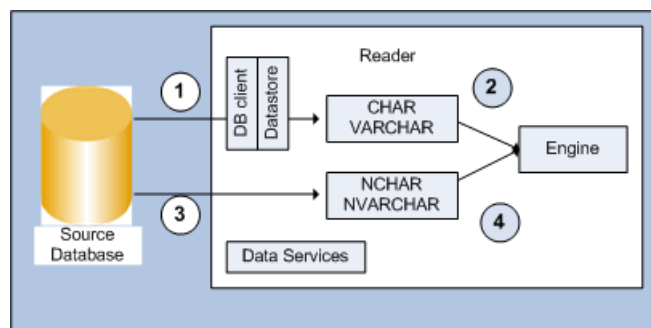
The code page you select in each locale used by a job determines whether transcoding occurs during job processing. Data Services automatically transcodes between different code pages when necessary to support complex, multi-language data management in a single job.

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.

### 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 Topics**

- [Column sizing](#)

### 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 Topics

- [Supported locales and encodings](#)

## 10.2.3 Guidelines for setting locales

### 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 BusinessObjects Data Services 4.1 > 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 Topics**

- [Minimizing transcoding](#)
- [Supported locales and encodings](#)
- [Example locale settings](#)

### **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.

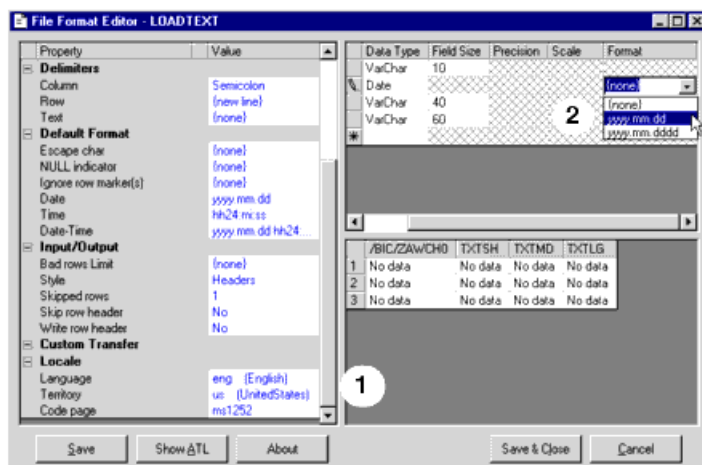
### Related Topics

- [Datastore](#)
- [XML encodings](#)
- [to\\_char](#)
- [to\\_decimal](#)
- [to\\_decimal\\_ext](#)

## 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:



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 Topics**

- [File format](#)
- [Designer Guide: File Formats, Number formats](#)

### 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 Topics**

- [Target XML files, messages, and templates](#)

### 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
Sybase ASE	OS Locale	LC_ALL	N/A
Sybase IQ	OS Locale	N/A	Character Set field
Teradata	ASCII	N/A	Session Character Set field

**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.

### 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.

## 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.

### 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.

### 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
<code>\$VALUE = 123.45</code>	Correct syntax.
<code>\$VALUE = 123,45</code>	Wrong syntax. Use a comma "," as a decimal separator inside Data Services only if the operating 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
<code>\$VALUE = '123,45'</code>	Correct syntax.
<code>\$VALUE = '123.45'</code>	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
<code>\$VALUE = '123.45'</code>	Correct syntax.
<code>\$VALUE = '123,45'</code>	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.

### 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.

## 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

### 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.

### 10.3.5 Column sizing

The number of bytes per character can vary from one code page to another. For example, the "ᄀ" 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.

## 10.4 Limitations of multi-byte support

There are several limitations to SAP BusinessObjects 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. EBCDIC 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.

## 10.5 Definitions

With regard to locales and multi-byte functionality associated with Data Services, refer to the following terms:

### 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.

### Encoding

The process of representing a code page character as one byte (single-byte encoding) or a sequence of bytes (multi-byte encoding).

### Language

Specifies the locale value for a human language.

**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.

**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.

**Single-byte**

An encoding or code page in which each character is represented by one byte.

**Territory**

Specifies the locale value for a geographical location (usually the country) where a locale language is used. The pairing 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

**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/IEC 10646. Data Services supports UTF-8 and UTF-16 Unicode transformation formats.

**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.

### 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.

## 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 `zh` and territory `cn` maps to Simplified Chinese, while the combination of `zh` and `tw` maps to Traditional Chinese.

### Supported languages

SAP BusinessObjects Data Services supports all three-letter language abbreviations specified in the ISO 639-2/T standard.

### Supported territories

SAP BusinessObjects Data Services supports all two-letter territory abbreviations specified in the ISO 3166-1 standard.

### Supported code pages

SAP BusinessObjects 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

Code page	Description			
	XML Encoding	Multi-byte	Vendor	Unicode Ver.
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			
	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

Code page	Description			
	XML Encoding	Multi-byte	Vendor	Unicode Ver.
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			
	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

# Python

## 11.1 Python

SAP BusinessObjects 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](http://www.python.org), also contains valuable information.

### Related Topics

- [About Python](#)
- [Create an expression with the Python Expression editor](#)
- [Defined classes and methods](#)
- [FIDataCollection class](#)
- [FIDataManager class](#)
- [FIDataRecord class](#)
- [FIProperties class](#)
- [FIPythonString class](#)
- [Python examples](#)

### 11.1.1 About Python

Python is an open-source, object-oriented scripting language. Python is installed with your SAP BusinessObjects 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:\userdata\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 Topics

- [Create an expression with the Python Expression editor](#)
- [Defined classes and methods](#)
- [FlDataCollection class](#)
- [FlDataManager class](#)
- [FlDataRecord class](#)
- [FlProperties class](#)
- [FlPythonString class](#)

## 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 Topics**

- [Smart editor](#)
- [To open the Python Expression editor from the User-Defined transform](#)
- [Write Python code](#)
- [Validate syntax](#)
- [Fix syntax](#)
- [Find and Replace](#)

### 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 Topics**

- [Smart editor](#)

### 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 Topics

- [Smart editor](#)

## 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 BusinessObjects 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 Topics**

- [Smart editor](#)
- [Python examples](#)

### **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

### **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.

### 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.

### 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	
FIData 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

Class	Object	Description	User-Defined Record Mode	User-Defined Collection Mode	Best Record (Match transform)
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

### 11.1.4 Defined classes and methods

The following table lists every SAP BusinessObjects 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)
FIData Collection	AddRecord		Yes	
	DeleteRecord		Yes	
	GetRecord		Yes	
	Size		Yes	
	Truncate		Yes	
FIData Manager	DeleteDataRecord		Yes	
	NewDataRecord		Yes	
FIData Record	GetField	Yes	Yes	Yes
	SetField	Yes	Yes	Yes
FIProperties	GetProperty	Yes	Yes	Yes

Class	Method	User-Defined Record Mode	User-Defined Collection Mode	Best Record (Match transform)
FIPython String	GetBuffer			Yes
	Set Buffer			Yes

## 11.1.5 FIDataCollection class

Use FIDataCollection 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 Topics

- [AddRecord](#)
- [DeleteRecord](#)
- [GetRecord](#)
- [Size](#)
- [Truncate](#)

### 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.
- 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)
```

## 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)
```

### 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,  <code>record = DataManager.NewDataRecord().</code>
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)
```



### 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()
```

### 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()
```

## 11.1.6 FIDataManager class

Use FIDataManager class methods when you want to create new records.

### Related Topics

- [DeleteDataRecord](#)
- [NewDataRecord](#)

### 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)
```

### 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 Topics**

- [About Python](#)

## 11.1.7 FIDataRecord class

Use Data Services-defined FIDataRecord class methods to manipulate existing individual records.

**Related Topics**

- [GetField](#)
- [SetField](#)

### 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	<p>In the Python Expression editor, use one of the input field variables.</p> <p>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.</p>

**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.

## 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.

### 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 Topics

- [GetProperty](#)

#### 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
```

## 11.1.9 FIPythonString class

Use the FIPythonString methods to customize your data processing. With these methods, you can create a Best Record operation in the Match transform.

**Related Topics**

- [GetBuffer](#)
- [SetBuffer](#)

### 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()
```

### 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
```

### 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
<p>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.</p>	<pre>record.SetField(u'SOURCE',u'TRC')</pre>
<p>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 upper-name.</p>	<pre>name = record.GetField(u'name') uppername = name.upper() record.SetField(u'uppername',unicode(uppername)) del name del uppername</pre>
<p>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.</p>	<pre>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</pre>
<p>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).</p>	<pre>field = record.GetField(u'firm') firm_length = field.strip().zfill(2) record.SetField(u'firm_length',unicode(firm_length)) del field del firm_length</pre>
<p>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.</p> <p>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.</p>	<pre>name_in = record.GetField(u'name') if len(name_in.strip()) == 0:     record.SetField(u'name',u'Valued Customer') del name_in</pre>

Example use	Sample Python code
<p>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.</p>	<pre>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</pre>
<p>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.</p>	<pre>groupnumber = record.GetField(u'groupnumber') groupnumberzeropad = groupnumber.strip().zfill(10) record.SetField(u'groupnumberzeropad', unicode(groupnumberzeropad)) del groupnumber del groupnumberzeropad</pre>
<p>In the User-Defined transform, append a field mapped to recordnum, and populate it with the record number.</p>	<pre>dct = locals() if dct.has_key('COUNTER'):     dct['COUNTER'] = dct['COUNTER'] + 1 else:     dct['COUNTER'] = 1 record.SetField(u'recordnum', unicode(dct['COUNTER']))</pre>
<p>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.</p>	<pre>dct = locals() if dct.has_key('COUNTER'):     dct['COUNTER'] = dct['COUNTER'] + 1 else:     dct['COUNTER'] = 1 recordnum = str(dct['COUNTER']).zfill(10) record.SetField(u'recordnum', unicode(recordnum)) del recordnum</pre>
<p>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 UserGroupA 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.</p>	<pre>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()) &gt; -1:     record.SetField(u'user_group', u'UserGroupA') elif ugb.find(user_code.strip().upper()) &gt; -1:     record.SetField(u'user_group', u'UserGroupB') del user_code del uga del ugb</pre>

### 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
<p>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.</p> <p>In this example, the syntax str specifies the type of split and the syntax 3-part specifies how to split.</p>	<pre>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</pre>
<p>Data is input in an input.account field (mapped 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.</p>	<pre>from flscansplit import ScanSplit account = record.GetField(u'account') type = [u'JT/WROS', u'JT WROS', u'JTWROS', u'JT/TEN', u'JT TEN', u'JT TEN', u'JT/TIC', u'JT TIC', u'JT TIC', 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</pre>

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'JT TEN', 'u'JT/TIC', 'u'JT TIC', 'u'JT TIC', '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
<p>Data is input with a field <code>gen.phone</code> 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.</p> <p>In this example, the Best Record strategy returns a <code>True</code> when the source is populated and the destination is empty (or else, it returns a <code>False</code>). At the end, the <code>GetField</code> method places the source data into the destination field, provided the Best Record strategy returns a <code>True</code>.</p>	<p><b>Best Record strategy:</b></p> <pre>Source = SRC.GetField(u'gen.phone') Destination = DST.GetField(u'gen.phone') if len(Source.strip()) &gt; 0 and len(Destination.strip()) == 0:      RET.SetBuffer(u'T') else:     RET.SetBuffer(u'F')</pre> <pre>del Source del Destination</pre> <p><b>Best Record Action:</b></p> <pre>RET.SetBuffer(SRC.GetField(u'gen.phone'))</pre>

## 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
<p>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.</p>	<pre> 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)) 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 </pre>

# Hadoop

Data Services can connect to Apache Hadoop frameworks including HDFS and Hive sources and targets. 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.

## 12.1 Prerequisites

Before configuring Data Services to connect to Hadoop, verify the following prerequisites:

- The Hadoop node has the Pig client installed, and if using Hive, the Hive client. Verify by logging on to the node and issuing pig and hive commands, which should invoke the respective interfaces.
- 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.

### Related Topics

- [Administrator's Guide: Server Management, To configure Job Servers \(Windows\)](#)
- [Administrator's Guide: Server Management, To configure Job Servers \(UNIX\)](#)
- [Designer Guide: Transforms, Text Data Processing transforms](#)
- [Configuring Hadoop for text data processing](#)

### 12.1.1 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 the 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.

#### Related Topics

- [Entity Extraction transform](#)
- [HDFS file format](#)

## 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 Topics

- [HDFS file format](#)
- [File format](#)
- [Designer Guide: File formats](#)



## 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, message broker, and SNMP communication**.
2. In the Administrator, add, configure, and start an adapter instance.
3. In the Designer, add and configure a Hive adapter datastore.

### 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, message broker, and SNMP communication**.

Use the Data Services Administrator to add and configure an adapter to connect to Hive.

1. Log in to Data ServicesManagement 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, enter 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 Topics

- [Administrator's Guide: Server Management, To configure Job Servers \(Windows\)](#)
- [Administrator's Guide: Server Management, To configure Job Servers \(UNIX\)](#)
- [Management Console Guide: Administrator, Adapters](#)
- [Designer Guide: Datastores, Adapter datastores](#)

## 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, message broker, and SNMP communication**.
- Add, configure, and start a Hive adapter instance.

Use the Data ServicesDesigner 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**, **Port number**, 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 Topics

- [Designer Guide: Datastores, Adapter datastores](#)
- [Datastore](#)
- [Hadoop Hive Adapter Source](#)
- [Hadoop Hive Adapter Target](#)

## 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 Topics**

- [Hadoop Hive Adapter Target](#)



## Reserved Words

### 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	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



## 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 database 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 Resource Planning (ERP), Customer Relationship Management (CRM), 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 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 categories: 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.



# Index

"DSN-less" connection 1137  
\\ See backslash 982  
|| See concatenation operator 257

## A

abbreviate  
    locality 599  
    USPS primary name 599  
abs function 790  
accept inexact postcode move 607  
access server 1137  
Access Server  
    real-time jobs 147  
    starting real-time jobs 150  
adapter 1137  
adapters  
    datastores, setting locale values for 1088  
    documents 106  
    message functions 144  
    outbound messages 145  
    source options 151  
    target options 165  
add a Data\_Transfer transform to a data flow 305  
AddRecord() 1110  
Address Accuracy report  
    Canada 1051  
address components 509  
Address Directory 579  
Address Geo Directory 579  
address geocoding 423, 444  
Address Information Code Summary 1029  
address line alias 599  
address lines  
    match minimum 611  
Address Quality Code Summary 1029  
address range window 611  
Address SHS Directory 579  
Address Type Summary 1031  
Address Validation Summary 1033  
ADDRINFOCODEDATA 1028  
ADDRINFOCODESUMMARY 1029  
ADDRSTATUSCODEDATA 1030  
ADDRTYPESUMMARY 1031  
ADDRVALIDATESUMMARY 1033  
Administrator 1137  
advanced, query editor 710  
after-image 1137

aggregate functions  
    avg (average) 793  
    count 799  
    description 777  
    max 897  
    min 898  
    sum 940  
AL\_SP\_ERRMSG 977  
AL\_SP\_RETCODE 977  
AL\_USAGE 1016  
alias 1137  
Allow merge or upsert 179, 189, 193  
ancestor/decendent relationships, metadata 1019  
ANKLink 1137  
annotation 1137  
annotations, description 27  
ANSI SQL-92 varchar 257  
append private mail boxes 599  
application 1137  
array fetch size, in SQL transform 732  
arrow icon 703  
assign postcode2 not DPV validated 607  
assign with input locality 599  
assign with input postcode 607  
assignment options 596  
Associate transform 370, 372  
    association options 372  
    best record 372  
    fields 384  
    group prioritization 372  
    group statistics 372  
    options 371  
    post-match processing 372  
    unique ID 372  
association matching 1137  
attributes  
    defined 23  
    for built-in repository objects 1013  
    table, retrieving 943  
Attunity Streams  
    data types, converting from 260  
audit point 1137  
Australia, report options 468  
automapping  
    content type 368  
avg function 793

## B

backslash (\\) 982  
batch job 1137  
batch jobs  
    description 28  
    objects in 28  
    recovering automatically 28  
    sources for 151  
batch project 1137  
before-image 1137  
best record 1137  
best record options 373, 557  
best record priority 1137  
best record summary 1058  
Best Record Summary 1057  
best record summary report 1057, 1066  
Binary Large Object 1137  
blank penalty 1137  
blank spaces in strings 982  
blob 1137  
blob data type 252  
blueprint 1137  
blueprints  
    downloading 365  
break group 1137  
break group options 524, 649  
break group statistics 1058, 1059  
    individual 1076  
break group strings  
    top and bottom ten 1060  
break key 418, 1137  
broker 595  
Broker and List Administrator 1045  
built-in functions 780  
bulk loading 1137  
    template tables and 209

## C

caching  
    Date\_Generation transform output 307  
    empty string  
        in lookup\_seq function 875  
    lookup function 866  
    lookup\_ext function 869  
    lookup\_seq function 875

- Row\_Generation transform output 730
    - source files 155
    - SQL transform output 732
    - Table\_Comparison transform 346
    - trailing blanks
      - in lookup\_seq function 875
  - calling reusable objects 22
  - Canada engine 459
  - Canada, options 459
  - candidate selection 523
    - User-Defined transform 649
  - candidate selection options 526, 652
  - capitalization 599
  - Case transform 695
    - editor 695
    - options 696
  - CASS 1137
  - CASS report options 609
  - CASS statistics 1046
  - catch functions 820, 821
  - catch object. See try/catch blocks 37
  - category 477, 616
  - CDC 1137
  - CDC checkpoint 1137
  - CDC data
    - sorting 333
  - CDC datastore 1137
  - CDC subscription 1137
  - ceil function 798
  - census data 423
  - central repository 1137
  - centroid geo directory 579
  - cgeo.dir 579
  - change options, Z4 615
  - changed-data capture 1137
    - source table options 152
  - City Addition 509
  - City directory 1137
  - City Directory 579
  - cityxx.dir 579
  - classes 1109
  - classification 1137
  - Cleansing Package option (Data Cleanse) 388
  - client/server 1137
  - COBOL copy book
    - data types, converting from 261
  - code pages 1079
    - defined 1095
    - Job Server restriction 1095
    - support in Data Services 1085
    - unsupported 1095
  - Coding Accuracy Support System 1137
  - column attributes, XML Schema 222
  - column icon 703
  - column mappings for imported tables, metadata 1002
  - columns
    - creating rows from 337
    - names in expressions 980
    - sizing 1094
  - columns in tables, imported into repository, metadata 1000
  - combine
    - ignore gaps 611
    - preserve gaps 611
  - Combine data sets 700
  - combine multilines 599
  - combine overlapping ranges 611
  - combine overlapping ranges, Canada 462
  - command line option, USA ShowA 691
  - command line options, USA ShowL 693
  - comments 237, 701
  - common 416
  - Common target table options 169
  - Common Warehouse Model 1137
  - company name certified 609
  - compare buffer 1137
  - compare table options 555
  - component 1137
  - concat\_date\_time function 799
  - concatenation
    - varchar data type and 257
  - conditional 1137
  - conditionals
    - description 48
    - using stored procedures 974
  - configurations, transform 359
  - connection string 1137
  - connections
    - to ODBC 83
  - consider moves within months 588
  - constant values, assigning to numeric data 1092
  - contact details
    - NCOALink 595
  - contact level 595
  - content model for DTDs 109
  - content objects 370, 387, 423, 506, 517, 577, 647
    - downloading 365
  - content type 1137
    - list of available 368
  - contribution value 1137
  - conversion
    - data type 287, 288, 289
    - decimal separator 1092
  - conversion functions 842
    - extract\_from\_xml 826
    - interval\_to\_char 842
    - julian\_to\_date 855
    - load\_to\_xml 862
    - long\_to\_varchar 865
    - num\_to\_interval 901
    - to\_char 944
    - to\_date 947
    - to\_decimal 948, 949
    - varchar\_to\_long 954
  - convert
    - native data types 286
  - converting data types 258, 286
  - count function 799
  - country ID options 446
  - Country ID transform 385
  - country name, country ID 446
  - country standardization options 446
  - criteria
    - individual 1076
  - criteria information
    - match contribution report 1076
  - CSLSTATS 1034
  - currency formats 1095
  - current schema 701
  - current schema, change 706
  - current\_configuration function 801
  - current\_system\_configuration function 802
  - custom ABAP program 1137
  - custom function 1137
    - delete an existing 965
    - edit an existing 964
    - replicate an existing 965
  - Custom Function tab 963
  - custom functions
    - creating 962, 963
    - using stored procedures 974
  - custom options
    - User-Defined transform 653
  - customer ID 584
  - customer information 584
  - customer option descriptions 584
  - Customer Service Log 1034
  - CWM 1137
  - CYRL 446
- ## D
- data
    - parsing 394
  - Data Cleanse transform 387
    - breaking input data 394
    - Cleansing Package option 388
    - Date options 403

- Data Cleanse transform (*continued*)
  - Input Word Breaker options 394
  - Options 394
  - Parser configuration options 405
  - Standardization options
    - Firm options 399
    - Gender options 398
    - Other options 399
    - Person options 395
- data cleansing fields
  - match prioritization 570
- data collection 418, 1137
- data flow 1137
- data flows
  - annotating 27
  - description 49
  - executing only once 51
  - execution, suspending 931
  - reserved words 1133
  - See also batch and real-time jobs 49
  - targets for 165
- data outputs 736
- data record 1137
- Data Services engine 1137
- Data Services repository 1137
- Data Services scripting language 979, 992
- Data Services service 1137
- data set 1137
- data set filters example 714, 758
- data sets, comparing 346
- data source name 1137
- data transformation 1137
- data transformations, transform objects 212
- data transforms
  - See also transforms 212
- data transport 1137
- data type 1137
  - date 287
  - datetime 287
  - decimal 288
  - double 288
  - int 289
  - varchar 289
- data type conversion 287, 288, 289
- data types 245
  - and column sizing 1085
  - blob 252
  - converting 258, 286
  - date 246
  - datetime 248
  - decimal 248
  - defined 245
  - double 249
- data types (*continued*)
  - int (integer) 250
  - interval 250
  - list of 245
  - long 251
  - national character-set support 1085
  - native 286
  - number, promotion of 283
  - numeric 254
  - real 255
  - stored procedure conversions 972
  - time 255
  - timestamp 256
  - varchar 257
  - XML Schema, types in 230
- data types, numeric 1092
- data validation 1137
- Data\_Transfer transform 296, 305
  - General tab 297
  - Options tab 301
  - Post-Load Commands tab 303
  - Pre-Load Commands tab 303
  - target options 297, 302
- database client locale
  - overriding 1090
- database datastores 60
- database functions 856
  - key\_generation 856
  - sql 932
- database link 1137
- database links, importing 53
- Database Management System 1137
- databases, connecting to 52
- DataConnector 1137
- dataflow\_name function 802
- datastore 1137
  - See also targets 52
- datastore configuration 1137
- Datastore Editor, dialog options 53
- datastore options
  - DB2 63
  - HP Neoview 66
  - Informix 68
  - MySQL 73
  - Netezza 77
  - Oracle 91
  - Sybase IQ 96
  - Teradata 101
- datastore\_field\_value function 803
- datastores 60
  - description 52
  - editor dialog options defined 53
  - importing database links 53
  - locale options 1088
  - ODBC 83
  - See also sources 52
- date arithmetic 258
- date data type 246, 287
- date functions 806
  - add\_months 791
  - concat\_date\_time 799
  - date\_diff 804
  - date\_part 804
  - day\_in\_month 806
  - day\_in\_week 807
  - day\_in\_year 808
  - fiscal\_day 828
  - isweekend 853
  - julian 855
  - last\_date 857
  - month 900
  - quarter 907
  - sysdate 941
  - sys\_time 942
  - week\_in\_month 956
  - week\_in\_year 957
  - year 962
- Date options (Data Cleanse) 403
- date\_diff function 804
- Date\_Generation transform 307
  - example 308
  - options 307
- date\_part function 804
- dates
  - converting to fiscal day 828
  - last in month, determining 857
- datetime data type 248, 258, 287
- day\_in\_month function 806
- day\_in\_week function 807
- day\_in\_year function 808
- db\_database\_name function 812
- db\_owner function 813
- db\_type function 809
- db\_version function 810
- DB2
  - data types, converting from 264
  - data types, converting to 264
  - SQL for, parameterizing 169
- DB2 datastore options 63
- DB2 target table options 179
- DBMS 1137
- debug mode 1137
- debugging
  - expressions 988
  - scripts 988
- decimal data type 248, 288
- decimal separator 1092
- decode function 813
- decrypt\_aes function 815
- decrypt\_aes\_ext function 816
- default configuration 53
- definitions of objects, changing 22

degree of parallelism 1137  
 DeleteDataRecord() 1114  
 DeleteRecord() 1111  
 deleting  
   data flows 49  
   data flows from real-time jobs 149  
   datastores 52  
   DTDs 114  
   file formats 123  
   log files 139  
   template table before loading 209  
   transforms 212  
   XML Schemas 233  
 delimited files  
   field names 123  
 delimiter 509, 1137  
 Delivery Point Barcode 1137  
 Delivery Point Validation 1137  
 delivery sequence file 415  
 DELSTATS 416  
 describing a function 963  
 design phase  
   real-time jobs using DTD formats,  
     improving 114  
   real-time jobs, testing 150  
   template tables, using for 209  
   XML messages, testing 219  
 Designer 1137  
 destination protection  
   unique ID 383, 567  
 DI\_ERRORACTION 742  
 DI\_ERRORCOLUMNS 742  
 diacritical character 1137  
 dictionary 1137  
 directional 1137  
 directional style 599  
 directory  
   DELSTATS 416  
 directory files, USA 579  
 directory, checking for 828  
 disable certification  
   CASS 607  
 disable certification, Canada 459  
 disambiguation 1137  
 discrete field 1137  
 discrete line format 1137  
 distinct rows 713  
 distinct tab 758  
 distinct, query clause 716  
 document objects 106  
 DOP 1137  
 double data type 249, 288  
 double quotation marks (") 980  
 DPBC 1137  
 DPV 1137  
 DPVLACSLINKSUMMARY 1035

drill down 1137  
 driver record 1137  
 dsf.dir 416  
 DSF2 415, 418  
   license information 417  
   output fields 421  
   walk sequence 416, 417  
 DSF2 Augment path 579  
 DSFAUGMENTSTATS 1037  
 DSFSEQUENCESTATS 1038  
 DSN 1137  
 DTD  
   ambiguous 114  
   attributes 108  
   description 106  
   error checking 115  
   See also XML messages 218  
   supported components 109  
 dual address 596, 1137  
 dual names 1137  
 dynamic input fields 366

## E

e-mail  
   sending lines from logs 884, 935  
 Early Warning System 1137  
 editor  
   DTD 107  
   file format 123  
   file source 155  
   persistent cache source 156  
   table source 152  
   table targets 169  
   template table 209  
   XML file source 160  
   XML message source 161  
 editor, query 710  
 Effective\_Date transform 309  
   Editor 311  
   options 311  
 eLOT 1137  
 eLOT directory 579  
 elot.dir 579  
 embedded data flow 1137  
 embedded data flows  
   port, setting source as 151  
   port, setting target 165  
 empty string  
   in lookup function 866  
   in lookup\_ext function 869  
   in rtrim\_blanks function 919  
   in rtrim\_blanks\_ext function 920  
   in substr function 938  
 Empty strings 986  
 empty strings, rules 986

enable  
   DPV 596  
   eLOT 596  
   EWS 596  
   LACSLink 596  
   NCOALink 596  
   RDI 596  
   SuiteLink 596  
 enable geo only 607  
 enable LVR rule, Canada 459  
 enable parse only 607  
 enable reverse soundex search 596  
 enable rural rule, Canada 459  
 enable suggestion lists 607  
 enable suggestion lists, Canada 462  
 encoding, defined 1095  
 encodings, setting for XML 1090  
 encrypt\_aes function 818  
 encrypt\_aes\_ext function 819  
 engine, Canada 459  
 engine, global address 464  
 engines  
   USA 469  
 engines, global address cleanse 458  
 Enhanced Line of Travel 1137  
 enterprise application 1137  
 enterprise resource planning system  
   1137  
 Entity Extraction transform 766  
   input fields 771  
   output fields 772  
 environment functions  
   get\_env 833  
   is\_set\_env 844  
   set\_env 930  
 Environment functions  
   get\_error\_filename 834  
   get\_monitor\_filename 836  
   get\_trace\_filename 836  
 ERP system 1137  
 error functions  
   descriptions 37  
   in catch script 39  
 error\_context function 820  
 error\_message function 821  
 error\_number function 821  
 errors  
   log files, description 143  
   log, example 143  
   parsing for validation 988  
   reserved words 1133  
 error\_timestamp function 821  
 escape characters  
   pushdown\_sql function 905  
   strings 982  
 escape characters in scripts 982

EWS 1137  
 EWS directory 579  
 ewsymmdd.dir 579  
 examples  
   Python 1121  
 exception 1137  
 exceptions  
   generating manually 908  
   list of 37  
 exec function 822  
 execute only once 51, 215  
 execution  
   properties for a batch or real-time  
     job 28  
     suspending 931  
 execution properties for jobs 30  
 execution status, stored procedures  
   977  
 expression 1104, 1137  
 expression editor 237  
 expressions 980  
   columns in 980  
   conversion of data types 283, 732  
   debugging 988  
   embedding in strings 983  
   evaluating during execution 988  
   functions in 983  
   in decode function 813  
   in ifthenelse function 839  
   null values in 257, 839, 986  
   operators in 984  
   stored procedures 974  
   stored procedures in 983  
   tables in 980  
 external processing updating list 588  
 extract\_from\_xml function 826  
 extraction options  
   common 766  
   dictionaries 769  
   languages 767  
   processing options 767  
   rules 770

## F

fail, validation 736  
 fault code 1137  
 fault codes 685  
 field category 477, 616  
 field delimiter 509  
 fields  
   Associate transform 384  
     prioritization fields 570  
   Geocoder 428  
   Match transform 570  
   prioritization 570

fields (*continued*)  
   USA Regulatory Address Cleanse  
     transform 616  
 file format 1137  
 file formats 41  
   description 123  
   editor modes 123  
   locale options 1089  
   properties 123  
   See also sources 116, 123  
   See also targets 116  
 file\_exists function 828  
 files  
   checking for 828  
   source options 155  
   targets, setting as 167  
 find  
   query transform 717  
   XML\_Map transform 762  
 finding match candidates 523, 649  
 firm 509  
 fiscal\_day function 828  
 FIDataCollection 1110  
 FIDataManager 1114  
 FIDataRecord 1116  
 floor function 829  
 FIProperties class 1118  
 FIPythonString 1119  
 Forward Sortation Area 1137  
 from tab, query transform 713  
 from, query editor 710  
 FSA 1137  
 function 1137  
 function call  
   editing 780  
 function wizard 780  
 function, browse for 242  
 function, search for 243  
 functional area 1137  
 functions  
   availability 778  
   built-in 780  
   built-in and imported metadata  
     1019  
   contrasted with transforms 777  
   custom 962, 965  
   descriptions 781  
   editing function call 780  
   expressions, adding to 779  
   expressions, using in 983  
   list of 781  
   multi-byte and locale support for  
     1092  
   object description 136  
   Python 1101  
   scope 777

functions (*continued*)  
   sources of 136  
   wizard 778  
 functions, nested 736

## G

gather 1137  
 gather statistics per data source 578  
 gen\_row\_num function 832  
 gen\_uuid function 832  
 gender 1137  
 generate report data 424, 445, 578  
 Generate report data option  
   Match transform 517  
 generated field 1137  
 geo directory, centroid 579  
 GEO mode 596  
 GEO\_ASSIGN\_LEVEL 1054  
 GEO\_INFO\_CODE 1055  
 Geocoder  
   about 423, 444  
   directories 444  
   fields 428  
     input 428  
     output 432, 439  
   options 425  
   repository tables 1026  
   sample transform 444  
 geocoder report and analysis 424  
 Geocoder transform  
   information codes 441  
 Geocoder transform tables 1053  
 get\_domain\_description function 832  
 get\_env function 833  
 get\_error\_filename function 834  
 get\_file\_attribute function 834  
 get\_monitor\_filename function 836  
 get\_trace\_filename function 836  
 GetBuffer() 1119  
 GetField() 1116  
 GetProperty 1118  
 GetRecord() 1112  
 Global Address Cleanse  
   repository tables 1026  
   statistics table 1028, 1029, 1030,  
     1031, 1033, 1051  
 global address cleanse engines 458  
 global address cleanse report options  
   445  
 global address cleanse transform  
   input fields 480  
 Global Address Cleanse transform  
   information codes 675  
   quality codes 684  
   status codes 679



- global address engine 464
- global address, engine 458
- global suggestion list 1137
- Global Suggestion List
  - about 505
- Global Suggestion List transform 505
  - Engines option group 507
  - Output\_Fields option group 513
- Global Suggestion Lists transform
  - Input\_Fields 512
  - Options option group 508
- Global Suggestion Lists transform
  - options 506
- group by tab 715, 759
- group by, query clause 716
- group by, query editor 710
- group forming 523
  - candidate selection options 526, 652
  - User-Defined transform 649
- group prioritization 523
- group statistics options 377, 561
- group statistics output fields 570

## H

- hierarchies, flattening 314
- Hierarchy\_Flattening transform 314
  - error conditions 323
  - options 315
- high match rate expectancy 588
- History\_Preserving transform 323
  - editor 324
  - options 324
- host name 1137
- host names, job execution, returning 838
- host\_name function 838
- HP Neoview
  - data types, converting from 263
  - data types, converting to 263
- HP Neoview datastore options 66
- HP Neoview target table options 184
- hybrid format 1137

## I

- ifthenelse function 839
- impact and lineage analysis 1137
- import 1137
- importing
  - database links 53
  - stored procedures 971
- importing metadata, data type
  - conversions 259
- include unused address line data 599

- index function 840
- index, columns in 162
- Info Code Sample Report 1028
- information codes
  - Geocoder transform 441
  - Global Address Cleanse transform 675
- Informix
  - data types, converting from 266
  - data types, converting to 266
- Informix datastore options 68
- Informix target table options 188
- init\_cap function 841
- inner join example 721
- inner joins 717
- inner table, joins 717
- input fields
  - dynamic 366
  - global address cleanse 480
  - walk sequencer 419
- input fields, Country ID 385
- input schema 701, 713
- input schema, query transform 703
- input schema, search 702, 747
- input source 1137
- input source match statistics 1066
- input source options 520
- input source statistics 1068
- installation, locale and multi-byte data
  - support 1087
- int (integer) data type 250
- int data type 289
- interactive debugger 1137
- interface 1137
- intersource match 1137
- interval data type 250, 258
- interval\_to\_char function 842
- is\_set\_env function 844
- is\_valid\_date function 845
- is\_valid\_datetime function 846
- is\_valid\_decimal function 848
- is\_valid\_double function 849
- is\_valid\_int function 850
- is\_valid\_real function 850
- is\_valid\_time function 851
- isempty function 852
- ISO codes 655
- isweekend function 853
- iterative functions 777

## J

- Japan engine 458
- job 1137
- Job Server 1137
  - code page restriction 1095

- Job Server (*continued*)
  - locale 1087
- job\_name function 854
- jobs
  - annotating 27
  - description 147
  - execution statistics for 1021
  - name of, returning 854
  - properties, default versus run-time 28
- join conditions example 714
- join example
  - mixed joins 724
- join rank 1137
- join, inner 721
- join, left outer 721
- join, outer 729
- join, query clause 716
- joins 717
- julian function 855
- julian\_to\_date function 855

## K

- key icon 703
- key\_generation function
  - description 856
- Key\_Generation transform 329
  - editor 329
  - options 329
- keywords for scripts 990, 991

## L

- LACS 1137
- LACSLink directory path 579
- language pack 1079
- last\_date function 857
- lastline 1137
- lastline components 509
- lastline match minimum 611
- lastlines match minimums, Canada 462
- latitude 423
- LATN 446
- LDU 1137
- left outer join example 721
- legacy system 1137
- length function 859
- library path 1101
- license-controlled feature 1137
- licensed-controlled feature 1137
- Licensee
  - assigned ID 595
- licensee name 584
- licensee option descriptions 584
- line 1137



Line of Travel 1137  
 linked datastores 1137  
 list  
   name 609  
   owner 609  
 list ID 584  
 list processing mode 588  
 list processing objective 588  
 list summary report  
   input list 1067  
   input source 1068  
 literal function 860  
 load\_to\_xml function 862  
 Local Delivery Unit 1137  
 local variables, defining 963  
 locale 1092, 1137  
   code page sizing 1094  
   code page, defined 1095  
   code page, support 1085  
   default value support 1097  
   defined 1095  
   encoding, defined 1095  
   language, defined 1095  
   multi-byte support for 1092  
   options for datastores 1088  
   options for flat files 1089  
   options for XML targets 1090  
   override default 1085  
   setting for the Job Server, during  
     installation 1087, 1095  
   setting values, for best performance  
     1087  
   supported code pages 1097  
   supported languages 1097  
   supported territories 1097  
   terminology 1095  
   territory, defined 1095  
   using in Data Services 1081  
   values for 1097  
 locale selection 1083  
 locales  
   example settings 1091  
 locality 1137  
 Locality1 509  
 Locality2 509  
 Locality3 509  
 Locatable Address Conversion System  
   1137  
 log file  
   DSF2 augment 1037  
   location 584  
 log files, description 139, 143  
 log tab 139  
 logical operators example 714, 758  
 long data type 251  
 long\_to\_varchar function 865

longitude 423  
 lookup function 729  
   description 866  
 lookup functions  
   lookup 866  
   lookup\_ext 869  
   lookup\_seq 875  
 lookup table 1137  
 lookup\_ext function 869  
 lookup\_seq function 875  
 LOT 1137  
 LOT certification 609  
 lower function 878  
 lpad function 879  
 lpad\_ext function 880  
 ltrim function 881  
 ltrim\_blanks function 882  
 ltrim\_blanks\_ext function 883  
  
**M**  
  
 mail piece unit 1137  
 mail\_to function  
   description 884, 935  
 mailer address1-4 609  
 mailing list name  
   NCOALink 587  
 Map\_CDC\_Operation transform 331,  
   334, 337  
   editor 331  
   options 332  
   sorting CDC data 333  
 Map\_Operation transform 698  
   real data type restriction 255  
 mapped field 1137  
 mapping 713, 755, 1006  
 mapping tab, query transform 701  
 mapping, query editor 710  
 mapping, query transform 712  
 mapping, XML\_Map transform 754  
 mappings  
   computing 1005, 1007  
   setting 701  
 master record 1137  
 match 1137  
   input source options 520  
   output flag selection options 568  
 match contribution report 1059, 1060,  
   1076  
 match criteria 1137  
 match criteria options 535, 541  
 match criteria statistics 1074, 1075  
 match criteria summary 1061  
 Match Criteria Summary 1062  
 match criteria summary report 1074,  
   1075

Match Duplicate Sample Report 1064  
 match engine 517  
 match group 1137  
 match mapped fields 535  
 match pattern 736  
 match range 611  
 Match repository table 1057, 1058,  
   1059, 1060, 1061, 1062, 1064,  
   1065, 1066, 1067, 1068, 1069,  
   1071, 1073, 1074, 1075, 1076  
 match set 1137  
   match criteria 1074  
 Match set 517  
 match source statistics report 1065  
 Match Source Statistics Report 1065  
 Match Source Stats Summary 1064  
 match stats summary 1064  
 Match transform 516  
   best record 556  
   best record options 373, 557  
   break group options 524, 649  
   candidate selection options 526,  
     652  
   compare table options 555  
   group forming 523  
   group statistics 556  
   group statistics options 377, 561  
   group statistics output fields 570  
   match criteria options 535  
   output fields 570  
   post-match processing 556  
   prioritization 556  
   prioritization fields 570  
   prioritization options 378, 528, 562  
   unique ID 556  
   unique ID options 380, 564  
   unique ID output fields 570  
 Match transform options 517  
 match\_pattern function 886  
 match\_regex function 889  
 matching  
   candidate selection 523  
 matching business rule options 541  
 mathematical functions 790  
   abs (absolute) 790  
   ceil 798  
   floor 829  
   rand 909, 910  
   round 914  
   trunc 951  
 max function 897  
 max number address lines 611  
 max number address lines, Canada  
   462  
 max number last lines 611  
 max number lastlines, Canada 462

- MEDSTATS 1039
  - memory datastore 1137
  - memory table 1137
  - Merge transform 700
  - message functions 144
  - Meta Integration Model Bridge 1137
  - metadata 1137
    - auditing 995
    - data type conversions 259
    - external, imported tables 1010
    - external, primary/foreign key relationships 998
    - external, primary/foreign key relationships among tables 997, 1002
    - external, properties of imported tables 1009
    - external, source and target column mappings 1002
    - external, table column properties 999
    - external, table columns 1000
    - imported, tables and views supporting analysis 997
    - importing 259
    - internal, attributes for built-in objects 1013
    - internal, objects in object library 1011, 1012
    - internal, parent/child relationships 1019
    - internal(built-in) and external (imported) functions 1019
    - mapping types 1004
    - mappings, computing 1005, 1007
    - nested data 1007
    - nested notations 1007
    - notations, nested 1007
    - NRDM 1007
    - operational, execution statistics for jobs 1021
    - operational, execution statistics for transforms 1022
  - metadata repository tables and views, reporting 995
  - methods 1109
    - Python 1101
  - Metropolitan Statistical Area 1137
  - Microsoft Excel
    - data types, converting from 267
  - Microsoft SQL Server
    - data types, converting from 268
    - data types, converting to 268
  - Microsoft SQL Server target table options 189
  - MIMB 1137
  - mimimizing transcoding 1087
  - min function 898
  - miscellaneous functions 813, 839
    - dataflow\_name 802
    - datastore\_field\_value 803
    - decode 813
    - file\_exists 828
    - gen\_row\_num 832
    - get\_domain\_description 832
    - host\_name 838
    - ifthenelse 839
    - isempty 852
    - job\_name 854
    - nvl 902
    - pushdown\_sql 905
    - raise\_exception 908
    - raise\_exception\_ext 908
    - repository\_name 914
    - sleep 931
    - system\_user\_name 942
    - table\_attribute 943
    - truncate\_table 952
    - workflow\_name 961
  - Miscellaneous functions
    - current\_configuration 801
    - current\_system\_configuration 802
    - db\_database\_name 812
    - db\_owner 813
    - db\_type 809
    - db\_version 810
    - gen\_uuid 832
    - get\_file\_attribute 834
  - Mode option 648
  - mode, country ID 446
  - monitor, log file, description 142
  - month function 900
  - move effective dates 1039
  - move multiline data 599
  - MSA 1137
  - MTBREACTION 1057
  - MTBRINFO 1058
  - MTBRKGRP 1059
  - MTBRKGRPINFO 1060
  - MTCMPCRIT 1061
  - MTCRITDEF 1062
  - MTDUPESDATA 1064
  - MTGSSRCBYSRCSTS 1064, 1066
  - MTGSSRCSTS 1065
  - MTINFO 1066
  - MTINSRCBYSRC 1066
  - MTINSRCGRPINFO 1067
  - MTINSRCINFO 1068
  - MTINSRCMSRC 1069
  - MTINSRCSELECT 1071
  - MTINSRCSTATS 1073
  - MTKEYDEF 1074
  - MTPROCESS 1075
  - MTRULESRES 1076
  - multi-byte data support
    - character restrictions 1095
    - definitions 1095
    - description 1092
    - functions 1092
    - limitations 1095
    - troubleshooting 1094
  - multi-source group statistics 1069
  - multiline 1137
  - multiline field 1137
  - multiline update postcode 1-2 599
  - MySQL
    - data types, converting from 270
    - data types, converting to 270
  - MySQL datastore options 73
- ## N
- NAICS 584
  - NAICS code 595
  - naming conventions, stored procedures 977
  - NANP 1137
  - native data types
    - conversions 286
  - nchar data type 1085
  - NCOACERTIFICATIONDATA 1039
  - NCOALink 1137
    - contact details 595
    - option descriptions 584
    - PAF 591
    - processing options 588
    - report options 590
    - report output options 591
    - service provider options 593
  - NCOALink options 587
  - NCOALink Report 1039
  - NCOALink statistics 1034, 1045
  - NCOALink Summary Report 1044
  - NCOALINKSUMMARY 1044
  - nested data 1137
  - nested data, documents, using for 106
  - nested functions 736
  - nested schemas 705, 746
  - nested tables, DTD conversion to 109
  - Netezza
    - data types, converting from 272
    - data types, converting to 272
  - Netezza datastore options 77
  - Netezza target table options 191
  - NewDataRecord() 1115
  - noise word 1137

- non certified options
  - USA regulatory address cleanse transform 607
- non-equality join 729
- normal source 1137
- North American Industry Classification system (NAICS) 584
- North American Numbering Plan 1137
- NULL 418
- NULL values
  - allowed in data types 245
  - keyword for 257, 839, 986
  - replacing, function for 902
  - testing for 257, 839, 986
- NULL, checking for it 986
- NULLS and empty stings 986
- NULLS and empty strings
  - in scripts 986
  - Oracle treatment 257
- num\_to\_interval function 901
- number data types, converting 283
- numeric data types 254, 1092
- numeric North American Industry Classification System 595
- numeric value 1092
- nvarchar data type 1085
- nvl function 902

## O

- object 1137
- object classes 21
- object definition 1137
- object dependent 1137
- object library 1137
- object version 1137
- OBJECT\_ID 1025
- OBJECT\_KEY 1025
- objects 21, 1108
  - attributes 23
  - descriptions 23
  - list of 24
  - options 23
  - properties 23
  - provided in object library, internal metadata 1011, 1012
  - reusable 22
  - single-use 23
- obsolete schema 713, 755
- ODBC
  - data types, converting from 273
  - data types, converting to 273
  - datastore options 83
  - SQL for, parameterizing 169
  - SQL\_LONG data type 251
  - ODBC target table options 193
- operating system, sending commands to 822
- operation code 1137
- operation codes, describing row status 291
- operational dashboard 1137
- operators in expressions and scripts 984
- operators, condition 736
- optimizing
  - data flow execution 49
  - expressions 283, 732
- Option Editor 1137
- Option Explorer 1137
- option group 1137
- options
  - Associate transform 371
  - options, Australia reports 468
  - options, Canada 459
  - options, country ID 446
  - options, definition 23
  - options, Validation transform 736
- Oracle
  - comparing trailing blanks 346
  - data types, converting from 275
  - data types, converting to 275
  - empty string treatment 257
  - error messages, saving 977
  - LONG data type 251
  - SQL for, parameterizing 169
  - table name syntax 980
- Oracle datastore options 91
- Oracle target table options 193
- order by, query clause 716
- order by, query editor 710
- order by, query transform 716
- order by, XML\_Map transform 760
- other source 1137
- outbound messages 145
- outer joins 717, 729
- outer table, joins 717
- output
  - suggestion list 611
- output address language, Canada 459
- output fields 621
  - DSF2 421
  - Match transform 570
- output flag selection options 568
- output options
  - NCOALink 591
- output schema 705, 713
  - changing 701
  - elements 701
  - filling automatically 701
  - icons 701
  - non-current 701
- output schema (*continued*)
  - stored procedures in 975
- output schema, change 706
- output schema, search 702, 747
- output style 509
- Output tab 477, 616
- overlapping ranges
  - combine 611
- overloading 966
  - defined 969
- override default locale 1085

## P

- package, defined 969
- PAF 591
  - sign date 595
- PAFBALASTATS 1045
- parameter 1137
- parameter list, defining 963
- parameterized SQL 169
- parameters
  - added by Data Services 972
  - execution 30
  - execution status 977
  - information for stored procedure 972
  - omitted from stored procedures 972
  - requirements for combining SQL 975
  - unspecified 977
- parent/child relationships, metadata 1019
- parse only, Canada 459
- Parser configuration options (Data Cleanse) 405
- partition 1137
- pass, validation 736
- passenger record 1137
- pattern file 1137
- PeopleSoft
  - flattening hierarchical data 314
- Per Collection option 648
- Per Collection processing mode 1101
- Per Record option 648
- Per Record processing mode 1101
- persistent cache
  - source options 156
- persistent cache target table options 167
- pivot sets, different sizes 337
- Pivot transform 337
  - options 338
- platform ID 584
- NCOALink 587

PMB 1137  
 POI 444  
 POI textual search 423  
 point of interest 423, 444  
 post-match processing  
     best record 372, 556  
     group statistics 372, 377, 556, 561  
     prioritization 372, 556  
     unique ID 372, 556  
 postal code 1137  
 postal directory 1137  
 Postcode Directory 579  
 postcode move 1137  
 postcode no match search, Canada 459  
 postcode only search, Canada 459  
 postcode priority over street, Canada 459  
 Postcode Reverse Directory 579  
 postcode1 509, 1137  
 postcode2 1137  
 precision 248  
 preferred viewing locale 1079  
 preserve dual address order 599  
 preserve place names 599  
 primary address components 509  
 primary entry 1137  
 primary key 703  
 primary keys  
     Table\_Comparison transform 346  
 Primary Name1 509  
 Primary Names Available 509  
 primary side indicator 509  
 primary type style 599  
 primary/foreign keys for metadata 998  
 primary/foreign keys for tables, metadata 1002  
 print function 904  
 prioritization fields 570  
 prioritization options 378, 528, 562  
 Private Mail Box 1137  
 processing  
     first-class mail 588  
     package services mail 588  
     periodicals 588  
     standard mail 588  
 Processing Acknowledgement Form 1045  
 Processing Acknowledgment Form NCOALink 591  
 processing frequency 584  
 processing mode 648, 1101  
 processing options  
     NCOALink 588  
 product locale 1079  
 project 1137

projection 1137  
 projects, description 145  
 properties  
     definition 23  
     description 23  
     execution 28  
     file formats 123  
     for built-in repository objects 1013  
     of imported table metadata 1009  
     trace 33  
 property 1137  
 PSFORM3553DATA 1046  
 pushdown\_sql function 714, 758, 905  
 pushing operations to database, stored procedures 975  
 Python 1101  
     classes 1109  
     examples 1121  
     fixing syntax 1107  
     FIDataCollection 1110  
     FIDataManager 1114  
     FIDataRecord 1116  
     FIPythonString 1119  
     objects 1108  
 Python API tab 1106  
 Python Expression editor 1104  
     check syntax 1107  
     opening 1105  
     using 1106  
 Python library 1101

**Q**

quality codes 684  
 quarter function 907  
 query clauses 716  
 query editor 717  
 Query Editor 702  
 query editor tabs 710  
 query object 721, 724  
 query transform 702, 705, 706, 712, 713, 714, 716, 717, 1137  
     automatic schema remapping 701  
     compared to SQL SELECT statements 701  
     mapping tab 701  
 Query transform 701  
     compared to User-Defined transform 646  
     editor 702  
     output schema 701  
 query transform, input schema 701  
 query transforms  
     description 146  
     output schema, stored procedures in 976

query transforms (*continued*)  
     using stored procedures 975  
 query, transform 715

## R

raise\_exception function 908  
 raise\_exception\_ext function 908  
 rand function 909, 910  
 RDI directory path 579  
 real data type 255  
 real-time job 1137  
 real-time jobs  
     compare batch jobs 149  
     data flows in 149  
     description 147  
     metadata 149  
     objects in 148  
     sources for 151  
     testing target data 149  
     transactional loading 150  
     XML messages and pushdown\_sql 905  
 rearranging data. See Pivot transform 337  
 recover as a unit 214  
 recovery, automatic  
     recovery unit 214  
     steps retrieved during 28  
     try/catch block 37  
     try/catch block and 213  
 reference file 1137  
 region1 509  
 relational data 1137  
 remap 713, 755  
 remapping, automatic in query transform 701  
 Remote Function Call server 1137  
 Remote Function Call server interface 1137  
 replace\_substr function 911  
 replace\_substr\_ext function 912  
 report and analysis options 445  
 report and analysis, geocoder 424  
 report and analysis, USA 578  
 report options  
     NCOALink 590, 591  
 reports  
     best record summary 1058, 1066  
     match contribution 1059, 1076  
     match contribution report 1060  
     match criteria summary 1061, 1062, 1075  
     match criteria summary report 1074  
     match duplicate sample 1064  
     match source statistics 1065

- reports (*continued*)
  - match stats summary 1064
- reports, Australia 468
- reports, creating using SQL statements 995, 1022
- repository 1137
  - reports, create using SQL statements 995, 1022
  - storing object definitions in 22
- repository statistics tables
  - Match 1055
- repository table 1025, 1028, 1029, 1030, 1031, 1033, 1034, 1035, 1037, 1038, 1039, 1044, 1045, 1046, 1049, 1051, 1052, 1054, 1055, 1057, 1058, 1059, 1060, 1061, 1062, 1064, 1065, 1066, 1067, 1068, 1069, 1071, 1073, 1074, 1075, 1076
- repository tables
  - AL\_ATTR 1013
  - AL\_AUDIT 995
  - AL\_AUDIT\_INFO 996
  - AL\_HISTORY 1021
  - AL\_INDEX 997
  - AL\_LANG 1011, 1012
  - AL\_PCOLUMN 998
  - AL\_PKEY 999
  - AL\_SETOPTIONS 1014
  - AL\_USAGE 1015
- repository tables introduction 1023
- repository views
  - ALVW\_COLUMNATTR 999
  - ALVW\_COLUMNINFO 1000
  - ALVW\_FKREL 1002
  - ALVW\_FLOW\_STAT 1022
  - ALVW\_FUNCINFO 1019
  - ALVW\_MAPPING 1002
  - ALVW\_PARENT\_CHILD 1019
  - ALVW\_TABLEATTR 1009
  - ALVW\_TABLEINFO 1010
- repository\_name function 914
- request/acknowledge operation 1137
- request/reply operation 1137
- required information
  - DSF2 417
- requirements
  - optimizing stored procedure in query 975
  - stored procedures 966
- reserved words 1133
- retain pound sign in unit description 599
- retrieve move types 588
- return codes 590
- return type, defining 963

- reusable object 1137
- reusable objects
  - description 22
  - single definition 22
  - storing 22
- reverse geocoding 423
- Reverse Pivot transform, rows to columns 343
  - options 344
- revzip4.dir 579
- RFC server 1137
- RFC server interface 1137
- round function 914
- row types 291
- Row\_Generation transform 730
- rows
  - creating from columns 337
  - retrieving multiple 732
- rows to columns, See Reverse Pivot transform 343
- rpadd function 915
- rpadd\_ext function 916
- rtrim function 918
- rtrim\_blank\_ext function 920
- rtrim\_blanks function 919
- rule file 1137
- rule matching 1137
- rules, validation 736
- run as separate process
  - DSF2 walk sequencer 416

**S**

- sample scripts 992
- sample size 1137
- sampling rate 1137
- sampling rows 1137
- SAP applications
  - flattening hierarchical data 314
- SAP Community Network 365
- SAP functions
  - defining
    - sap\_openhub\_processchain\_execute function 923
    - sap\_openhub\_processchain\_execute 920
    - sap\_openhub\_set\_read\_status 923
- SAP HANA
  - data types, converting from 276
  - data types, converting to 276
- SAP HANA database datastore options 81
- scale 248
- schema out 705
- schema remap 713, 755
- schema remapping 712, 754

- schema, input 703
- schema, obsolete 713, 755
- schemas
  - current, finding in query editor 701
- schemas, nested 746
- script 1137
- script code, country ID 446
- Scripting language 963
- scripts
  - debugging 988
  - description 150
  - for catch 40
  - functions in 983
  - keywords 990, 991
  - operators in 984
  - purpose of 980
  - samples 992
  - stored procedures in 974
  - validation 242, 988
  - when to use escape characters 982
- search
  - query transform 717
  - XML\_Map transform 762
- search input/output schemas 702, 747
- search\_replace 925
- search\_replace function 925
- search/replace, query editor 710
- second generation delivery sequence file 415
- secondary address components 509
- secondary information 1137
- segment 1137
- select input source statistics 1071
- SELECT statements, equivalent in Data Services 701
- select tab, query transform 713
- select, query editor 710
- selection 509
- SendRight Address Accuracy report 1049
- SENDRIGHTADDRACCURACY 1049
- SERPADDRACCURACY 1051
- server group 1137
- service provider options
  - NCOALink 593
- set\_env function 930
- SetBuffer() 1120
- SetField() 1117
- signature
  - added parameters 977
  - changes to 971
  - definition 971
  - support for Byte Order Mark 1093
  - viewing 971
- similarity score 1137
- single use object 1137



- single-byte data support 1079
  - single-use objects
    - description 23
  - site location
    - DSF2 417
  - Size() 1113
  - sleep function 931
  - smart editor 1137
  - Smart editor 237
  - Smart Editor
    - Python Expression editor 1104
  - smtp\_to, enabling 938
  - snowbird 1137
  - software version 609
  - source group 1137
  - source group statistics 1073
  - source record 1137
  - source statistics 1067
  - sources
    - description 151, 161
    - documents 106
    - embed flow port, setting 151
    - files 41, 116, 123
    - retrieving multiple rows 732
    - tables 162
    - template tables 209
    - XML files 216
    - XML messages 218
  - spatial search geocoding 423
  - SQL
    - function
      - escape characters 932
      - parameterizing 169
      - transform 732
  - sql function 932
  - SQL transform 732
  - Standardization options (Data Cleanse)
    - Firm options 399
    - Gender options 398
    - Other options 399
    - Person options 395
  - standardization options, USA 599
  - standardization, country 446
  - standardize assigned address 599
  - standardize unassigned address 599
  - standardized output fields
    - NCOALink 591
  - standards 1137
  - star schema 1137
  - stateless functions 777
  - statistics 1025, 1028, 1029, 1030, 1031, 1033, 1034, 1035, 1037, 1038, 1039, 1044, 1045, 1046,
    - statistics (*continued*)
      - 1049, 1051, 1052, 1054, 1055, 1066, 1075, 1076
      - break group strings 1060
      - input source 1067, 1068
      - Match break groups 1058, 1059
      - match criteria 1074
      - multi-source groups 1069
      - repository 1055
      - select input source 1071
      - source groups 1073
    - Status Code Sample Report 1030
    - status codes 679, 686
    - step 1137
    - stored procedure
      - generic example, using 967
    - stored procedures
      - calling 974
      - creating in DB2 968
      - creating in MS SQL Server 970
      - creating in Oracle 969
      - creating in SAP HANA 970
      - creating in Sybase ASE 970
      - data type conversions 972
      - defined 965
      - entering manually 977
      - execution status, monitoring 977
      - expressions, using in 983
      - importing 971
      - naming convention 977
      - omitted parameters 972
      - query output schema 975
      - query transforms, using in 975
      - requirements 966
      - signature 971, 977
      - SQL, combining with query 975
      - structure 972
      - viewing 971
    - storing reusable objects 22
    - street address 1137
    - string format 1092
    - string functions 840
      - index 840
      - init\_cap 841
      - length 859
      - lower 878
      - lpad\_ext 880
      - ltrim 881
      - ltrim\_blanks 882
      - ltrim\_blanks\_ext 883
      - print 904
      - replace\_substr 911
      - replace\_substr\_ext 912
      - rpadd 915
      - rpadd\_ext 916
      - rtrim 918
  - string functions (*continued*)
    - rtrim\_blanks 919
    - rtrim\_blanks\_ext 920
    - substr 938
    - upper 953
    - WL\_GetKeyValue 958
    - word 958, 960
  - strings
    - concatenating 257
    - converting to numbers 283
    - embedding expressions in 983
    - empty, with NULLS 986
    - replacing portions of 911, 912
  - sub data flows, Query transform 716
  - subordinate record 1137
  - substitution parameter 1137
  - substitution parameter configuration
    - 1137
  - substr function 938
  - suggestion list 1137
    - output 611
    - USA Regulatory Address Cleanse 611
  - suggestion list options 509
  - suggestion list options, Canada 462
  - sum function 940
  - summary report
    - best record 1057
  - supported countries 655
  - suppression source 1137
  - suspending execution 931
  - Sybase ASE
    - data types, converting from 278
    - data types, converting to 278
  - Sybase ASE target table options 198
  - Sybase IQ
    - data types, converting from 279
    - data types, converting to 279
  - Sybase IQ datastore options 96
  - Sybase IQ target table options 199
  - syntax
    - checking Python 1107
    - fixing Python 1107
  - syntax for scripts 979
  - sysdate function 941
  - system functions
    - exec 822
    - mail\_to 884, 935
  - system\_user\_name function 942
  - sysptime function 942
- T**
- table 1137
  - table attributes, XML Schema 222
  - table\_attribute function 943

Table\_Comparison transform  
 description 346  
 options 347  
 real data type restriction 255

tables  
 attributes, retrieving 943  
 comparing 346  
 description 162  
 imported into repository, metadata 1010  
 index, viewing 162  
 loading in single transaction 150  
 names in expressions 980  
 partitions, viewing 162  
 repository 1025, 1028, 1029, 1030, 1031, 1033, 1034, 1035, 1037, 1038, 1039, 1044, 1045, 1046, 1049, 1051, 1052, 1054, 1055, 1057, 1058, 1059, 1060, 1061, 1062, 1064, 1065, 1066, 1067, 1068, 1069, 1071, 1073, 1074, 1075, 1076  
 retrieving multiple rows 732  
 source options 152  
 source options for CDC 152  
 statistics 1057

target 1137

target options  
 Data\_Transfer transform 297, 302

target table options  
 common 169  
 DB2 179  
 HP Neoview 184  
 Microsoft SQL Server 189  
 Netezza 191  
 ODBC 193  
 Oracle 193  
 Sybase ASE 198  
 Teradata 201

targets 41, 123  
 description 165, 207  
 documents 106  
 embedded data flows, setting port 165  
 files 167  
 outbound messages 145  
 persistent cache table options 167  
 table options 169  
 tables 162  
 template tables 209  
 Writer migrated from Data Quality 209  
 XML files 216  
 XML files, options 207  
 XML messages 218  
 XML messages, options 207

targets (*continued*)  
 XML template 234

TDPID 1137

template tables  
 description 209  
 target options 209

Teradata  
 data types, converting from 281  
 data types, converting to 281

Teradata datastore options 101

Teradata Director Program ID 1137

Teradata target table options 201

territory 1137

testing  
 real-time jobs 150  
 real-time jobs with DTD formats 114  
 template tables, using for 209

textual search 423

third-party Python library 1101

thread 1137

time data type 255, 258

time functions 942

times, arithmetic including 258

timestamp data type 256, 258

to\_char function 944

to\_date function 947

to\_decimal function 948, 949

tokenization 1137

total\_rows function 950

trace log files  
 data type errors in 259  
 description 140

trace properties for a job 33

trailing blanks 982  
 in decode function 813  
 in ifthenelse function 839  
 in lookup function 866  
 in rtrim function 918  
 in scripts 982  
 in substr function 938  
 in Table\_Comparison transform 346  
 in varchar operations 257

transactional loading 150

transcode  
 defined 1095  
 minimizing 1087  
 round-trip conversion conflicts 1094

transform 1137

transform configurations  
 list of 359

transform options, USA standardization 599

transform, Country ID 385

transform, query 702, 705, 706, 712, 713, 714, 715, 716, 717

transform, USA Regulatory Address Cleanse 577

transform, Validation 736, 737, 739, 746

transform, XML\_Map 747, 754, 758, 759, 760, 762

transforms 323  
 Associate 370  
 Case 695, 696  
 Data Cleanse 387  
 Data\_Transfer 296  
 Date\_Generation 307  
 description 212  
 Effective\_Date 309  
 Entity Extraction 766  
 input fields 771  
 output fields 772  
 execution statistics, operational metadata 1022  
 Global Address Cleanse 444  
 Global Suggestion List 505  
 Hierarchy\_Flattening 314  
 Key\_Generation 329  
 list of 292  
 Map\_CDC\_Operation 331  
 Map\_Operation 698  
 Match 516  
 Merge 700  
 Pivot 337  
 Query 701  
 reserved words 1133  
 Reverse Pivot, rows to columns 343  
 Row\_Generation 730  
 SAP applications 314  
 See also individual transform names 292  
 SQL 732  
 User-Defined 646

translation 1079

trunc function 951

truncate\_table function 952

Truncate() 1113

try/catch block 1137

try/catch blocks  
 description of catch 37  
 description of try 213  
 exceptions, table listing 37  
 real-time jobs, using with 37

**U**

Unicode 1101  
 support 1079, 1085

Unicode (*continued*)  
 support for Byte Order Mark 1093  
 support for UTF-16 1079  
 support for UTF-8 1079  
 UTF16 1085  
 unique ID 383, 567  
 unique ID options 380, 564  
 unique ID output fields 570  
 unique identifier 1137  
 unique record 1137  
 unit description 599  
 unit description, Canada 459  
 unsuccessful jobs, examining 140  
 unsupported data types 260, 278, 279  
 upper function 953  
 US Addressing Report 1035  
 US Regulatory Locking Report 1052  
 USA directories 579  
 USA engine 458, 469  
   suggestion list options, USA 470  
 USA Regulatory Address Cleanse  
   repository tables 1026  
   statistics 1037, 1038  
   statistics table 1028, 1029, 1030,  
     1031, 1033, 1034, 1035, 1039,  
     1044, 1045, 1046, 1052  
 USA Regulatory Address Cleanse  
   transform 577  
   fault codes 685  
   fields 616  
   options 578  
   status codes 686  
   Transform\_Performance option  
     group 582  
 USA report and analysis 578  
 USA ShowA 691  
 USA ShowL 693  
 USA transform options, standardization  
   599  
 use cases 1016  
 use USPS locality abbreviation 599  
 use USPS primary name abbreviation  
   599  
 user interface  
   translated 1079  
 User-Defined transform  
   break group options 524, 649  
   candidate selection 649  
   candidate selection option 526, 652  
   common usages 646  
   compared to Query transform 646  
   custom options 653  
   description 646  
   group forming 523, 649  
   options 648  
 USPS Form 3553 1046

USPS license information 586  
 USREGULATORYLOCKING 1052  
 UTF-8 1079, 1087, 1088  
 UTF16 1079

## V

validate icon 242, 963  
 validating  
   Python syntax 1107  
 validation functions 845  
   is\_valid\_date 845  
   is\_valid\_datetime 846  
   is\_valid\_decimal 848  
   is\_valid\_double 849  
   is\_valid\_int 850  
   is\_valid\_real 850  
   is\_valid\_time 851  
 validation rule, add 739  
 Validation transform options 736  
 Validation, transform 736, 737, 739,  
   746  
 validation, viewing errors 988  
 var-graphic data type 1085  
 varchar data type 257, 289  
 varchar\_to\_long function 954  
 variables  
   assigned to empty strings 986  
   with NULLS and empty strings 986  
 verifying  
   Python syntax 1107

## W

walk sequence  
   DSF2 416, 417  
   output 421  
 walk sequencer 415  
   input fields 419  
 web service request 1137  
 web services 1137  
 week\_in\_month function 956  
 week\_in\_year function 957  
 weighted scoring 1137  
 WHERE clause  
   checking for NULLs 986  
   dynamic, creating 905  
   real data type restrictions 255  
 where tab, query transform 714  
 where tab, XML\_Map transform 758  
 where, query editor 710  
 while loops, description 213  
 WL\_GetKeyValue function 958  
 word function 958  
 word\_ext function 960  
 work flow 1137

work flows  
   annotating 27  
   description 214  
   executing only once 215  
   execution, suspending 931  
   name, retrieving during job 961  
   recovering as a unit 214  
   reserved words 1133  
 work flows in real-time jobs, data flows  
   in 149  
 workflow\_name function 961  
 workspace 1137  
 Writer migrated from Data Quality  
   target options 209

## X

XML files  
   creating without DTD or XML  
     Schema 234  
   description 216  
   source options 160  
   target options 207  
 XML messages  
   content model for DTDs 109  
   description 218  
   in real-time jobs 148  
   source options 161  
   target options 207  
 XML Schema  
   attributes mapped to column  
     attributes 226  
   column attributes 222  
   data type mappings 230  
   description 220  
   elements mapped to attributes 224  
   elements not supported 229  
   error checking 234  
   import rules 227  
   nested table attributes 222  
 XML source, extracting data from a  
   single column 826  
 XML target, loading data to a single  
   column 862  
 XML template 234  
 XML\_Map transform 747, 754, 758,  
   760, 762  
   Advanced tab 753  
   configure mappings 753  
   DISTINCT tab 753  
   Find tab 753  
   GROUP BY tab 753  
   input schema 746  
   Iteration Rule tab 753  
   Mapping tab 753  
   ORDER BY tab 753



XML\_Map transform (*continued*)  
  output schema 746, 750  
  WHERE tab 753  
XML\_Map, transform 758, 759  
XML\_Pipeline transform 356  
  editor 356

## Y

year function 962

## Z

Z4 change 615  
Z4 Change directory 579

Z4Change 1137  
z4change.dir 579  
ZCF 1137  
zcfxx.dir 579  
ZIP City File 1137  
ZIP plus 4 1137  
ZIP+4 615  
zip4us.dir 579  
zip4us.shs 579

